UPC Commons

Motor para videojuegos en HTML5, basados en tiles

Autor: Rodolfo Núñez del Río
Director/Ponente : Lluís Solano Albajés

21/01/2014

Índice

Índice de figuras

0. Prólogo

En este documento los autores explicaremos todos los detalles del proyecto que hemos estado trabajando durante seis meses.

Cabe comentar los siguientes aspectos importantes del proyecto:

- Hasta el punto 6 de la memoria es común en ambos documentos.
- Las definiciones y explicaciones que se encuentran en la documentación sobre la evolución de los videojuegos, los géneros y subgéneros, y los elementos de los videojuegos son basadas en las experiencias de los autores.

1. Introducción

1.1. Descripción del proyecto

En este proyecto se presenta el desarrollo de un *game engine* utilizando el lenguaje de etiquetas HTML5 y basado en *tiles*, dando prioridad a la facilidad de uso por parte de creadores no relacionados con el desarrollo de videojuegos.

Un ***game engine*** es un conjunto de rutinas de programación que facilitan el diseño y desarrollo de videojuegos. Existen actualmente un gran número de *game engines* para diferentes plataformas y algunos de ellos centrados en el desarrollo de un tipo específico de videojuegos.

El *game engine* provee un motor de renderizado de gráficos en 2D o 3D, también puede incluir un motor físico, detector de colisiones, sonido, animación, inteligencia artificial, etc.

Actualmente las grandes desarrolladoras de videojuegos crean sus propios *game engines* para agilizar el desarrollo de sus videojuegos. Existen empresas que se dedican exclusivamente al desarrollo de *game engines*, como Unity u Ogre.

Una ***tile*** es una unidad básica de construcción. Tiene como una de sus características que se repite varias veces en el escenario. Un videojuego basado en *tiles* es un juego que utiliza baldosas o bloques iguales como uno de los elementos fundamentales del juego. Los juegos tradicionales de tablero basados en tiles utilizan estos bloques para crear el escenario, dando así una gran cantidad de posibilidades. Los videojuegos basados en tiles, se basan en este mismo objetivo, a

partir de bloques iguales, normalmente con forma cuadrada, podemos crear un escenario formando una secuencia de imágenes como si fuera una cenefa. A estas tiles se les da unas propiedades y un comportamiento, para completar las normas del juego.

1.2. Objetivos

El objetivo es el desarrollo de un *game engine* que pueda ser utilizado online desde cualquier dispositivo capaz de ejecutar un navegador compatible con las nuevas tecnologías HTML5, hemos querido aprovechar las posibilidades que nos ofrece HTML5 para desarrollar videojuegos para navegador.

Los objetivos propuestos para el desarrollo de este proyecto han sido:

- El desarrollo de un editor que permita la fácil creación de videojuegos para navegador
- Dividir el editor gráfico en dos bloques, uno que tratará la física y otro que tratará la decoración visual.
- Hacer un sistema amigable de selección de archivos, para introducirlos en el editor y posteriormente en el juego, de una forma simple y rápida para el usuario.
- Dotar el editor de las herramientas necesarias para que el usuario pueda trabajar cómodo a la hora de crear los escenarios o pantallas de la aplicación.
- Capacidad de ser ejecutado en un navegador web sin necesidad de algún pluguin o instalación previa.
- Editor dotado de memoria que nos permita almacenar en nuestro dispositivo el trabajo que hemos realizado.
- Editor capaz de leer los archivos que nosotros generamos.
- Una estructura intuitiva para la fácil ampliación de los elementos básicos que proporciona el editor, el usuario podrá introducir nuevos elementos, con muy pocos pasos.
- El desarrollo de una serie de rutinas que permitan el funcionamiento de dicho juego.
- Desarrollo de una serie de rutinas básicas que permitan el desarrollo fácil de unos pocos géneros básicos de videojuegos
- Creación de varios perfiles que permiten que el personaje jugable se comporte de diferente forma.
- Rutinas de "físicas" para otorgar un comportamiento realista del juego, como gravedad o colisiones.
- Una Inteligencia Artificial muy básica que rige el comportamiento de algunos personajes.
- Implementación de un algoritmo de cálculo eficiente de trayectorias para enemigos y jugadores controlados mediante ratón.

- Una estructura intuitiva para la fácil ampliación del código.
- Capacidad para cargar escenarios y aplicaciones creados por el editor.
- Capacidad para ejecutarse en distintos dispositivos, PC, Tablet, dispositivos móviles y potencialmente cualquier dispositivo capaz de conectarse a Internet y ejecutar un navegador web compatible con HTML5

1.3. Estructura del proyecto

El proyecto se ha dividido en dos bloques claramente diferenciables, por una parte el editor, desarrollado por Rodolfo Núñez del Río, y por otra el conjunto de rutinas que controlan la física y la inteligencia del juego, desarrollada por Víctor Manuel Agüero Requena.

Cuando ambos bloques trabajan juntos forman el Game Engine, capaz de generar un escenario o pantalla por el editor y ser ejecutado por el motor.

2. Evolución hasta *Game Engines*

2.1. 1950 - 1960: Antecedentes

Durante las décadas de 1950 y 1960, tras la segunda guerra mundial, se había comprobado sobradamente el potencial que tenían aquellas enormes máquinas, antecesores de los ordenadores actuales, y se intuía que podrían llegar a hacer grandes cosas con ellos. Gracias a investigadores como Alan Turing se había llegado a la conclusión que los campos que más posibilidades entrañaban eran el de la interacción hombre-máquina y el de la Inteligencia Artificial, y los videojuegos tenían la posibilidad de probar los avances de ambos campos manteniendo a la vez el interés de los sujetos testados.

Hay varios debates respecto a cuál fue el primer videojuego de la historia, el objetivo de este documento no es aportar pruebas ni argumentos para descubrir cual fue, pero es interesante hacer un repaso a los dispositivos y softwares que se disputan ese puesto.

Empezaremos por *Cathode ray tube amusement*[1] o «Dispositivo de entretenimiento de tubos de rayos catódicos», patentado el 25 de enero de 1947. Éste dispositivo quería simular el

[1] Actualmente existe un software que emula el funcionamiento del dispositivo, puede verse una demostración en el siguiente video: http://www.youtube.com/watch?v=k_WUb-1C010

lanzamiento de misiles y estaba inspirado por los radares utilizados durante la segunda guerra mundial.

Se utilizaba una serie de circuitos analógicos para controlar el haz de un tubo de rayos catódicos y la posición de un punto en una pantalla. El jugador disponía de un tiempo para posicionar el punto para que se superpusiera al dibujo de un avión y pulsar un botón para disparar al avión. Si el jugador fallaba, el rayo se desenfocaba para simular una explosión.

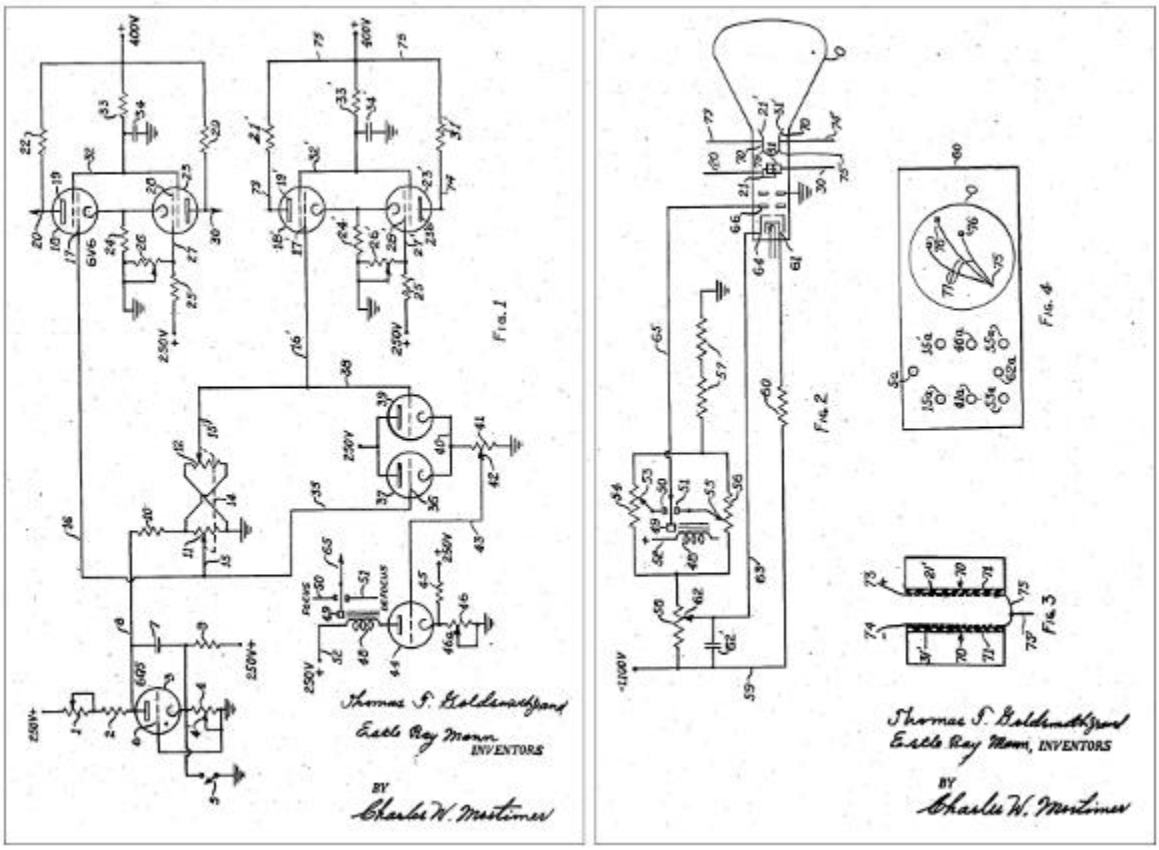


Figura 1. Esquema de Cathode ray tube amusement

El siguiente elemento que podríamos considerar como un antecedente a los videojuegos viene de mano de Alan Turing, que en 1951 escribió un programa capaz de jugar al ajedrez como ejemplo de inteligencia en los computadores. El programa requería demasiada potencia de cálculo para ser ejecutada en ninguna máquina de la época, por lo que el mismo Turing debía simular el funcionamiento del programa, tardando más de hora y media en completar un movimiento.

El siguiente ejemplo en este campo lo tenemos en NIMROD, presentada el 5 de mayo de 1951 en el festival británico, que fue la primera máquina creada con el único propósito de jugar al NIM, un juego que consiste en que dos o más jugadores retiren fichas por turnos, con el objetivo de ser el

que retire la última ficha. NIMROD fue desmontada un tiempo después para utilizar sus piezas en otros proyectos.

El 30 de julio de 1951 Christofer Strachey desarrolló una simulación del juego de damas para *Pilot ACE*, uno de los primeros ordenadores construidos en el Reino Unido para Laboratorio Nacional de Física.

En el 1952 Alexander S. Douglas escribió *OXO*, un programa de juego de tres en raya como ejemplo de interacción humano-máquina. Fue escrito para EDSAC y se controlaba mediante un dial rotatorio telefónico. Fue el primer juego en utilizar gráficos digitales.

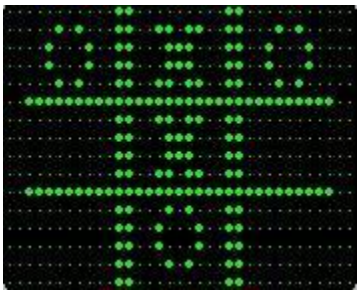


Figura 2. Tres en raya

No se vuelve a saber de más ejemplos de entretenimiento electrónico de este tipo hasta el 1958, año en el que William Higinbotham creó *Tennis for Two* con el objetivo de que sirviera de demostración para el día de visita del Brookhaven National Laboratory, financiada por el Departamento de Energía de los Estados Unidos. *Tennis for Two* utilizaba gráficos vectoriales para representar el juego.

Entre 1959 y 1961 el MIT desarrolló una colección de juegos para TX-0, una máquina experimental. Entre ellos destacaban *Mouse in the Maze*, en el que el jugador usaba un lápiz óptico con el que añadir muros y queso, para a continuación soltar al "ratón", que se movía por el laberinto para encontrar todos los bits de queso.
Otro juego de la colección era *Tic-Tac-Toe*, un juego de tres en raya en el que el jugador utiliza el lápiz óptico para introducir los círculos y las cruces en el tablero.

El siguiente ejemplo viene también del MIT. En 1961 los estudiantes Martin Graetz, Steve Rusell y Wayne Wiitanen crearon Spacewar! en una DEC PDP-1, una de las entonces llamadas minicomputadoras. Éste es considerado el primer videojuego de disparos y fue utilizado como prueba para los nuevos sistemas PDP-1, ya que ponía a prueba todos los aspectos del *hardware*.

Figura 3. Spacewar! ejecutándose en una PDP-1

En 1966 Ralph Baer empezó a trabajar para desarrollar un videojuego que pudiera ser visualizado en un televisor. En 1967 Baer y su asociado crearon el primer juego que se puede considerar, sin ningún atisbo de dudas, un videojuego, así como también la primera videoconsola. En 1972, *BrownBox*, el último prototipo de esta consola fue lanzada al mercado por Magnavox con el nombre de *Odyssey*.

2.2. 1970: Primeras recreativas

A finales de la década de los 60, Bill Pits, un estudiante de la Universidad de Stanford tuvo la idea de crear una versión de *Spacewar!* que funcionase con monedas para su explotación comercial, pero debido al precio del *hardware* necesario para hacerla en aquel momento la idea no resultaba económico, y no fue hasta el 1971 en que, junto a Hugh Tuck, creó la empresa *Computer Recreations Inc*. y se dispuso a crear la máquina. Tras tres meses y medio de trabajo crearon *Galaxy Game*, la primera máquina arcade de la historia. La máquina fue relativa exitosa, pero no resultaba rentable, por lo que tuvieron que hacer una segunda versión en la que un solo ordenador PDP-11 se hacía cargo de hasta 8 máquinas. Esta máquina estuvo en servicio con bastante éxito desde 1972 hasta 1979, fecha en la que se retiró. Desde el año 2000 se exhibe en el *Computer Museum History Center de Mountain* View, California.

Figura 4. Galaxy Game, primera máquina arcade de la historia

Tan solo dos meses después del lanzamiento de *Galaxy Game*, salía al mercado *Computer Space*, otra conversión de *Spacewar!* desarrollada por Nolan Bushnell y Ted Dabney, fundadores de Syzygy Engineering. Las primeras pruebas comerciales de estas máquinas, instaladas en bares universitarios, tuvieron mucho éxito, por lo que Nutting Associates, decidió empezar la fabricación en masa de estos aparatos, convirtiendo de paso a Galaxy Game en la primera máquina arcade en ser fabricada en serie. Sin embargo, el resultado no fue el esperado, ya que el juego resultaba demasiado complicado para el público no universitario, por lo que al final la operación no resultó rentable. El 27 de junio de 1972 Bushnell y Dabney, por problemas legales, tuvieron que cambiar el nombre de su empresa a Atari.

Fue en esta época en la que Magnavox comercializó Odyssey, la primera videoconsola de la historia. En 1971 se empezó su fabricación en cadena, y en abril de 1972 comenzó su distribución. Magnavox también distribuyó el primer accesorio, un rifle de plástico, y diez videojuegos todos de venta por separado. Pese a los errores de marketing, Odyssey consiguió vender cerca de 13000 unidades en la campaña de navidad, un éxito que atrajo la atención de varios emprendedores, entre ellos Nolan Bushnell.

Figura 5. Máquina recreativa de PONG

El 24 de mayo de 1972 Nolan Bushnell estuvo presente en la presentación de Odyssey, ese día tuvo la oportunidad de jugar a *Ping-Pong*, uno de los juegos que salieron junto a la videoconsola. Bushnell vió que el juego podía ser mejorado y explotado, por lo que contrató a Alan Alcorn, un ingeniero de Ampex, que mejoró el control añadió marcadores y efectos de sonido, así nació *Pong*. La primera recreativa de Pong se probó en Andy Capp's Tavern, al día siguiente de su instalación, Alcorn recibió una llamada de los propietarios de Andy Capp's Tavern, diciendo que la máquina se había averiado. El error estaba producido por el depósito de monedas, que se hallaba repleto. Este hecho animó a Bushnell a ofrecer la producción de *Pong* a Nutting Associates, pero debido al fracaso de su anterior asociación, Nutting rechazó la oferta, con lo que Bushnell se vió obligado a encargarse de la fabricación y distribución de las máquinas.

Las primeras unidades se vendieron con facilidad, por lo que Bushnell decidió crear la máquina en masa, para ello contrató a un gran número de operarios de una oficina de desempleo, muchos de ellos ex presidiarios adictos a la heroína o al hachís, y pese a los problemas que esto provocó, Nolan pudo hacer frente a los pedidos de máquinas de *Pong*, cada vez mayores. A finales de 1974 las máquinas arcades generaban más de 250 millones de dólares anuales en beneficios. Quedaba demostrado que la industria del videojuego podía ser muy rentable.

Con el lanzamiento de *Pong*, Atari se convirtió en la líder indiscutible de la industria, mientras que las competidoras se limitaban a lanzar copias de *Pong*, Atari, libre de las restricciones que suponían los componentes mecánicos, susceptibles a averías, lanzaba juegos innovadores para la época, como *Gotcha*, *Quadrapong* o *Qwak!*, prácticamente inaugurando un nuevo género con cada lanzamiento.

En 1973 Bushnell creaba Kee Games, una filial dirigida por Joe Keenan y que fue presentada como la competencia de Atari, en un intento de burlar las leyes de industria, en diciembre de 1974 la relación entre ambas empresas salió a la luz, pero Kee Games siguió distribuyendo juegos hasta 1978, cuando sus máquinas empezaron a llevar el sello "*Kee Games, a wholly owned subsidiary of Atari, Inc*".

En 1975 Atari lanzaba Telegames Pong, su primera videoconsola doméstica, que permitía jugar a *Pong* en casa, fue un éxito que atrajo a un gran número de competidores.

También en 1975 salió al mercado *Gun Fight*, este arcade fue el primero en utilizar microprocesadores de propósito general, este no era solo un cambio en la fabricación del videojuego, sino que también implicaba que ahora no sería necesario hacer cambios en el

hardware cada vez que se quisiera cambiar algún aspecto del videojuego, no solo eso, sino que ahora los diseñadores de juegos pasaban a ser los programadores.

Los últimos juegos en no utilizar microprocesadores fueron *Death Race* de Exidy y *Breakout* de Atari.

Death Race, lanzado en 1976, consistía en conducir atropellando el mayor número de zombies posibles, como es de imaginar, este juego generó mucha polémica en su momento, y fue a consecuencia de este videojuego que se planteó la posible influencia de los videojuegos en la infancia.

Breakout, era el intento de Atari de convertir su exitoso Pong a un videojuego de un jugador. El primer prototipo fue encargado a Steve Jobs, a quien prometieron un incentivo de 100$ por cada chip que pudiera eliminar del circuito. Jobs ofreció a Steve Wozniak compartir el trabajo y las ganancias y Wozniak consiguió reducir el número de chips a 42. Bushnell pagó 5000$ a Steve Jobs, pero éste le dijo a Wozniak que solo le habían pagado 700$, por lo que le pagó solo 350$ y se quedó la diferencia.

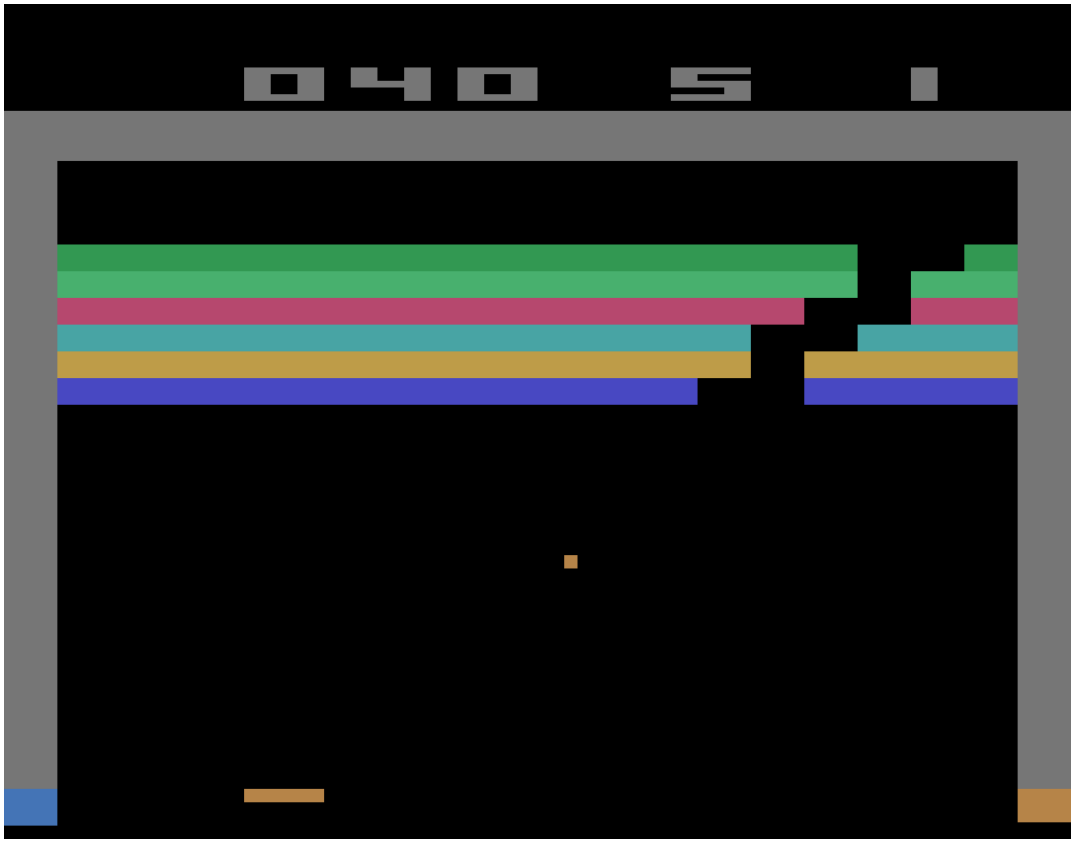


Figura 6. Breakout, último juego en utilizar tecnología TTL

En 1975 Will Crowther, un programador que trabajaba en el departamento de defensa de los Estados Unidos creó un pequeño programa en el que narraba una aventura en una cueva y el jugador decidía a dónde se dirigía y que hacía, con la intención de jugar con sus hijas. El programa se hizo muy popular dentro de la red ARPAnet y varios usuarios de la red fueron haciendo modificaciones. De esta forma nació Adventure, una de las primeros juegos conversacionales y el que se considera inaugurador del género de las aventuras.

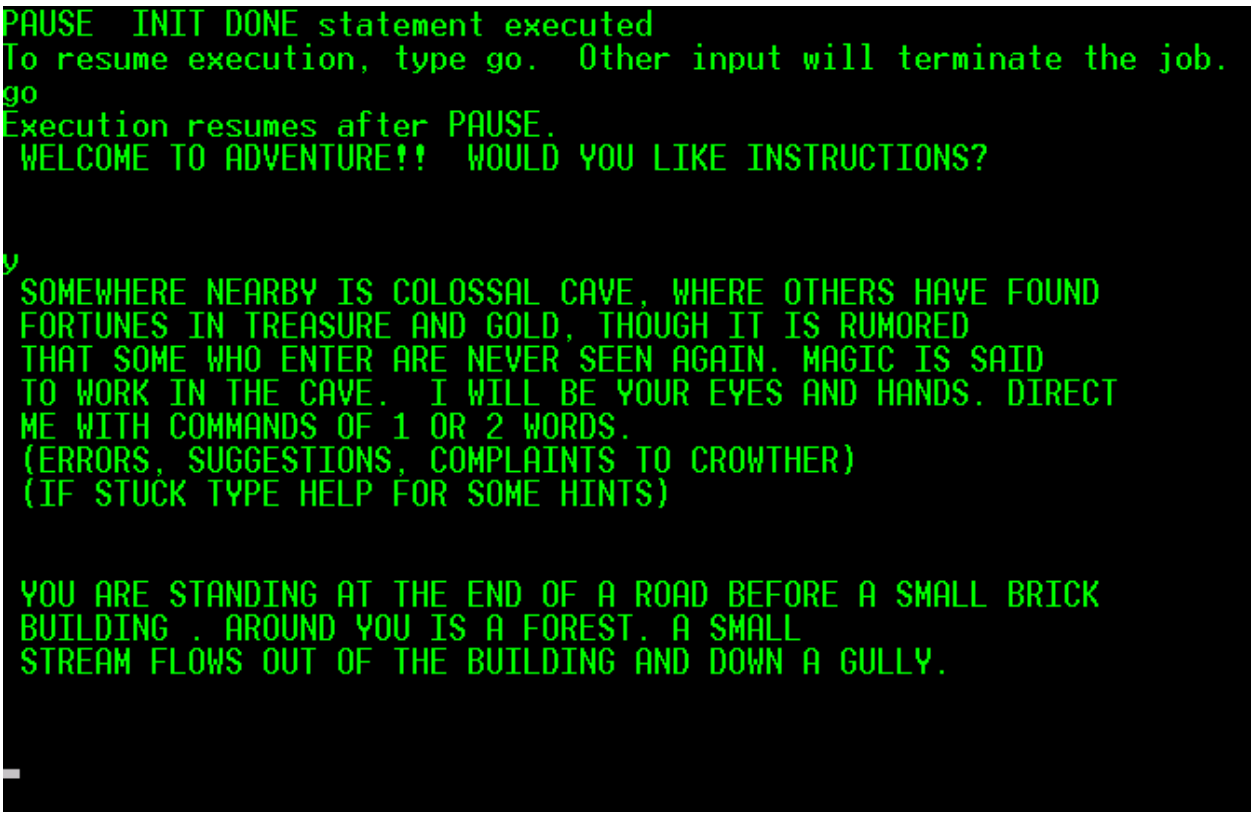


Figura 7. Adventure, de Will Crowthers

Fue en medidos de la década de los 70 cuando empezaron a salir los primeros microordenadores como Altair 8800 o Apple I, con un precio asequible para el gran público. Hasta ahora los ordenadores eran máquinas enormes y caras, solo al alcance de grandes empresas y departamentos gubernamentales, es por esta razón que en esta época tenían gran éxito los salones recreativos, pero los microordenadores permitían jugar con algunos de esos videojuegos desde casa. Durante esta época surgió la industria "casera" del videojuego que fueron el germen de algunas de las grandes empresas del futuro, como es el caso de Ken y Roberta Williams, que tras publicar *Mystery House* para Apple II, una adaptación de Diez Negritos de Agatha Christie y la primera aventura con gráficos, fundaron *Sierra Entertainment*, empresa que en el futuro sería una de las grandes distribuidoras de videojuegos norteamericana y una de las dos principales desarrolladoras de Aventuras Gráficas, junto con *LucasArts*.

Figura 8. Apple II

A principios de 1976 Atari se encontraba desarrollando el que sería uno de los proyectos por los que más se recordaría a la empresa en el futuro, se trataba de Atari 2600, la primera videoconsola en utilizar cartuchos intercambiables, y que en teoría permitiría alargar la vida útil de la máquina.

Figura 9. Primera portátil Auto Race

Debido a problemas financieros, Atari tuvo que retrasar el lanzamiento de la consola, y en aiagosto de 1976 salió al mercado Fairchild Channel F, una consola de una empresa rival que consiguió así adelantarse a Atari por primera vez, pero esta no fue la única consola que salió al mercado, a principios de 1977 se comercializaba RCA Studio II y no fue hasta el 1978, con la venta de Atari a Warner Bros, que Atari lanzó Atari 2600, que contó en su lanzamiento con un catálogo mayor que el de sus competidoras. El mercado de las videoconsolas domésticas ya estaba definido. Mientras tanto también hubo avances en el nicho de las videoconsolas portátiles, en 1976 Mattel sacaba Auto Race[2], la primera consola portàtil de la historia.

Tras la venta de Atari, se fue apartando a Bushnell de la dirección, hasta que en 1978 fue sustituido por Ray Kassar, que decidió

[2] Para una mayor comprensión de cómo funcionaba la videoconsola, recomendamos visualizar el siguiente video: http://www.youtube.com/watch?v=cfx4p4WMJ9s

enfocarse en el mercado de los ordenadores personales y terminando así la época dorada de Atari.

Mientras tanto, en Japón, Tomohiro Nishikado creó *Space Invaders* para Taito, que originalmente consistía en disparar contra tanques y aviones, pero que debido al éxito de Star Wars y a presiones de la empresa, acabó convirtiéndose en un juego de batallas espaciales. El juego fue un gran éxito, se convirtió a todos los formatos y se vendió por todo el mundo. El juego cosechó tal éxito, que en japón hubo escasez de monedas y el gobierno de Japón tuvo que aumentar el número de monedas en circulación.

2.3. 1980 -1990: Los 8 y 16 bits

A finales de la década de los 70 y principios de los 80 se vivía una auténtica fiebre por los videojuegos. Muchos programadores aficionados habían ganado pequeñas fortunas y empresas dedicadas al negocio del entretenimiento habían creado divisiones centradas en el desarrollo de videojuegos, como LucasArts o Walt Disney Pictures, aprovechando también las licencias de las películas que ellos mismos producían.

El uso de los gráficos vectoriales y el color pasaron de ser anecdóticos a convertirse en la norma e incluso algunas empresas se especializaron en el desarrollo de videojuegos que utilizaran y mejoraran el uso de estos recursos, como es el caso de Cinematronics.

Fue en este ambiente en el que se gestó uno de los videojuegos más populares de la historia, sino el más popular, se trata de Pac-Man, del japonés Toru Iwatani, que creó en un intento de atraer al público femenino al mundo de los videojuegos. El éxito fue nunca visto hasta la fecha, y aún hoy Pac-Man sigue produciendo grandes beneficios ya no en la venta de juegos, sino en merchandising.

Estados Unidos y Japón se habían establecido como grandes productoras de entretenimiento electrónico, pero Europa estaba muy por detrás y se había estancado en el papel de consumidor. Fue en 1980 cuando los aficionados europeos pasaron de consumidores a productores, con la salida de Sinclair ZX80, el primer microordenador personal de 8 bits, con los años le siguieron ZX81 y ZX Spectrum y la competencia lanzó al mercado máquinas de las mismas características entre las que destacan Amstrad CPC, también de origen europeo, Commodore 64, de una empresa estadounidense, y MSX, un estándar de origen nipón . Estas máquinas permitieron a los aficionados no solo jugar con videojuegos, sino también modificar el código de éstos e incluso

crear nuevos videojuegos. Fue gracias a estas plataformas que gente como los hermanos Richard y David Darling pudieron fundar Codemasters.

Figura 10. ZX Spectrum

Es interesante destacar el importante papel que desempeñó España como productor de *software* lúdico gracias a estas máquinas, con videojuegos como La Puga de Indescomp, la serie *Army Moves* de Dynamic y La Abadía del Crimen de Opera Soft.

Figura 11. Screenshot de La Pulga

Mientras, en Japón, visto el gran éxito que había tenido *Space Invaders*, muchas empresas se decidieron a utilizar el videojuego como vehículo para promocionar su cultura y su país a occidente e introducirse en el mercado estadounidense y europeo. Fue en este contexto en que Nintendo en 1983, una empresa productora y distribuidora de cartas y juguetes que buscaba diversificar el negocio se introdujo en el negocio de los videojuegos y acabó por lanzar al mercado Famicom, o Nintendo Entertainment System (NES), como sería conocida fuera de Japón, que contaba con juegos ya conocidos como *Donkey Kong*, a esta consola se remontan los inicios de sagas de videojuegos mundialmente famosas como *Super Mario*, *The Legend of Zelda*, *Dragon Quest* o *Final Fantasy*.

Mientras en Japón se gestaba lo que sería el principal mercado de videoconsolas del futuro, en Estados Unidos el mercado del entretenimiento electrónico se veía inmerso en una crisis, la gran cantidad de videojuegos que se lanzaban y la escasa calidad de muchos de ellos hizo que el público no tuviera confianza en los nuevos lanzamientos, y esto llevó a la quiebra a gran cantidad de empresas dedicadas al desarrollo de videojuegos. Y fue entonces, en 1985, cuando NES salió a la venta en el mercado estadounidense, el gran éxito que cosechó contribuyó a la salida de la crisis del sector en el país. Visto el éxito que tuvo NES, muchas compañías intentaron lanzar sus propias videoconsolas caseras de 8 bits, pero solo Sega, con su Master System, logró alcanzar cierto éxito.

Fue a mediados de los 80 que Sierra Entertainment, que se había iniciado en el negocio de los videojuegos con Mistery House, desarrolló *Adventure Game Interpreter* (AGI) para uso interno. AGI era un parser utilizado para facilitar el desarrollo de las aventuras gráficas de la compañía, y puede considerarse como uno de los primeros *Game Engines* de la historia. AGI fue utilizado en todas las aventuras gráficas de la compañía desde 1984 hasta 1988, momento en que fue sustituido por *Sierra's Creative Interpreter*, que aprovechaba mejor las capacidades de los ordenadores que había en el mercado en aquella época e incorporaba la utilización del ratón.

La década de los 90 supuso el fin de la generación de las videoconsolas de 8 bits, así como el nacimiento de su sucesora, la generación de 16 bits.

A principios de esta década triunfaron las aventuras gráficas de LucasArts (llamada entonces LucasFilms Games), entre las que destacan *Maniac Mansion* y *The Secret of Monkey Island*, que utilizaban el entonces novedoso motor *Script Creation Utility for Maniac Mansion* (SCUMM), motor especializado en el desarrollo de aventuras gráficas y uno de los primeros *Game Engines*, la

posibilidad de interactuar con el juego con el ratón en lugar de tener que escribir los verbos con el teclado fue revolucionaria. El motor SCUMM fue utilizado por la compañía hasta 1997.

También en estas fechas sacaba Sega su Mega Drive, conocida como Génesis en Estados Unidos, primera videoconsola de 16 bits y adelantándose así a Nintendo, que había sido la indiscutible controladora del mercado hasta el momento y no fue hasta 1992 que Nintendo sacó Super Nintendo, con lo que recuperó su liderazgo, Sega respondió lanzando Mega CD, pero no tuvo el éxito esperado.

Figura 12. Sega Megadrive

El retraso Nintendo en el mercado de las 16 bits se debía al desarrollo de Game Boy, sucesora de su serie de máquinas portátiles Game & Watch, diferentes compañías intentaron hacerle la competencia lanzando sus versiones de videoconsolas portátiles, como Atari con su *Lynx* o Sega con su *Game Gear*, pero ninguna fue capaz de competir con Game Boy, y finalmente Nintendo se hacía líder indiscutible del mercado de las videoconsolas portátiles.

2.4. 2000 - Actualidad: La popularización de los videojuegos

Llegados a este punto de la historia, es necesario hablar de los nuevos usuarios que surgieron con el tiempo y que esto provoca el cambio en el estilo de los juegos que van apareciendo.

A los jugadores que acostumbran a jugar a videojuegos se les llama videojugadores, o también con el anglicismo *gamer*. Un *gamer* es, un jugador que se caracteriza por jugar con gran dedicación e interés a videojuegos y por tener un conocimiento diversificado sobre estos.

También, aunque más ocasionalmente, se designa en español a estos jugadores habituales como *geeks* de los videojuegos, o frikis. Sin embargo cuando un jugador alcanza un nivel de adicción al juego se dice de él que es un ludópata. No obstante existen muchas más clases de jugador:

- **Casual**: (La mayor parte de jugadores se encuentran en este colectivo.)Se denomina así a aquel individuo que no está comprometido propiamente a conseguir todos los objetivos posibles en un juego dado. Además, estos jugadores no suelen emplear ni la mitad de tiempo que un Hardcore en un juego dado, caracterizándose, a nivel general, por jugar a muchos juegos, pero durante poco tiempo o en intervalos irregulares.

- **Regulares**: Muchas personas no consiguen encajar en la categoría de jugador hardcore, pero a su vez tampoco en jugadores casuales, para estos se asigna el término de jugadores regulares el cual se refiere a un jugador que juega de manera habitual, tiene ciertos conocimientos de los videojuegos, pero no busca un gran reto como los jugadores hardcore ni se esmeran tanto en ser los mejores en algún juego.

- **Hardcore**: Se caracteriza por ser un jugador que dedica grandes horas al día a jugar videojuegos. Busca mejorar constantemente y tener puntuaciones máximas, estos jugadores no quedan satisfechos con terminar un videojuego de manera habitual pues buscan siempre conseguir todo lo que se pueda en una alta dificultad y con grandes retos, también gustan de modos competitivos para demostrar sus habilidades. Estos jugadores son verdaderamente aficionados a los videojuegos, no buscan solo un medio de entretenimiento sino un reto o una aventura épica.

- **Gosu**: Que se caracteriza por ser un gamer con habilidades extraordinarias para jugar, y por ello, es considerado un jugador experto solo superado por máster.

- **Máster**: Experto total juega de todos los géneros y casi nunca es vencido.

- **Troll**: Solamente se mete a jugar para molestar a los demás.

Dada esta nueva distribución de usuarios por los videojuegos, compañías como Nintendo optaron por dirigirse a este público en concreto, produciendo una videoconsola de mucho menor coste, juegos más sencillos y baratos, pero orientados a este público que posteriormente iba a causarles muchos más ingresos. Así se creó la Wii de Nintendo.

No obstante las compañías en la actualidad siguen viendo el negocio en los *hardcore gamers*, ya que ellos no solo mueven dinero comprando el producto en si, sino con la cantidad de publicidad que pueden introducir en todos los eventos que organizan.

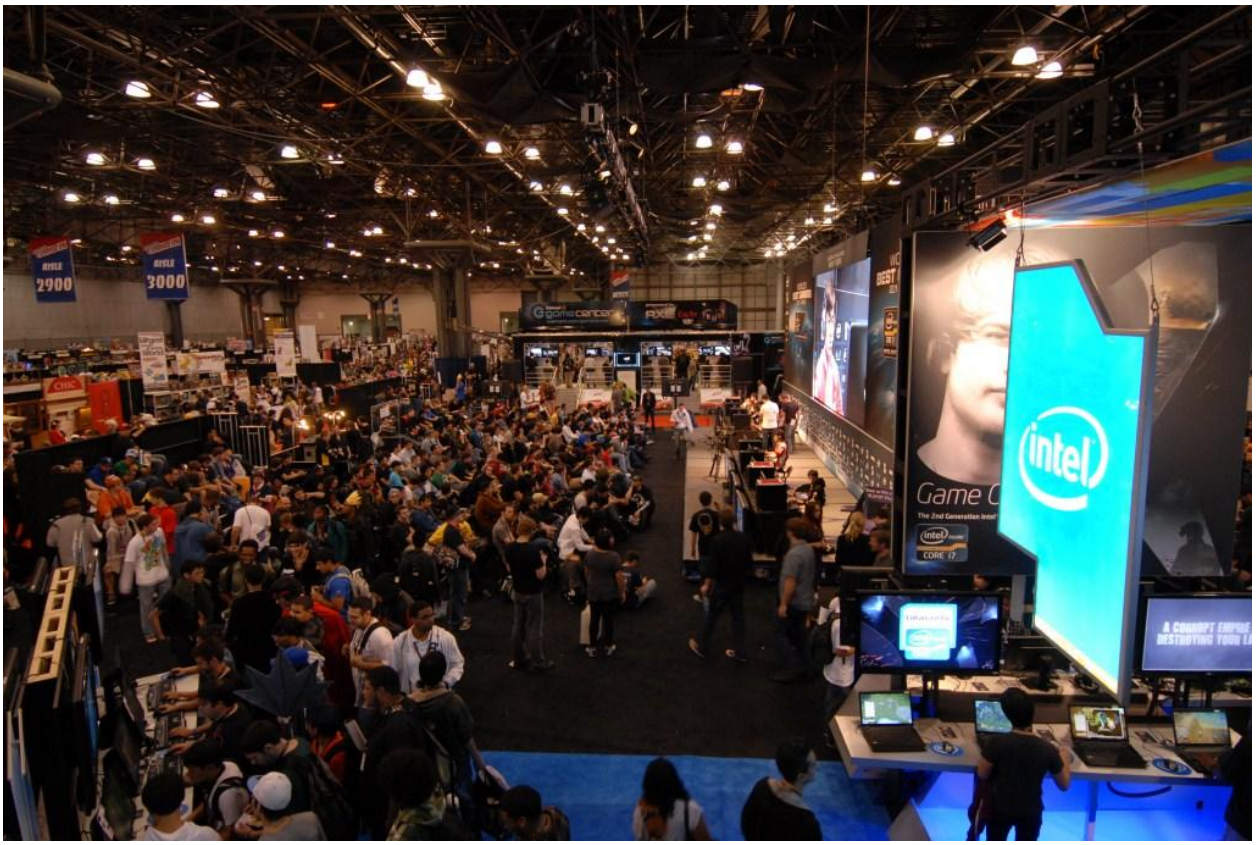


Figura 13. Imagen de una competición de League of Legends patrocinada por intel.

Gracias a todos estos cambios la industria del videojuego en la actualidad se ha transformado en una industria multimillonaria de dimensiones inimaginables pocos años antes. En 2009 la industria de los videojuegos era uno de los sectores de actividad más importantes de la economía estadounidense, y en países como España generaba más que la industria de la música y el cine juntos. El programador de videojuegos ya no era el aficionado a la electrónica que elaboraba prácticamente en solitario y con carácter artesanal y *amateur* sus programas, sino un profesional altamente cualificado que trabajaba con otros profesionales especializados en equipos de desarrollo perfectamente estructurados a menudo bajo el control directo o indirecto de grandes multinacionales.

La gracia de las videoconsolas siempre había sido su mayor potencia respecto a los ordenadores de sobremesa para ejecutar videojuegos. La Neo-Geo de SNK permitía al jugador disfrutar en su casa de la misma tecnología que exhibían las máquinas de los salones recreativos, pero era una consola muy cara sólo al alcance de unos pocos, un videojuego común podía costar a un jugador

¡30.000 de las antiguas pesetas!. El éxito de la *Playstation*, *SegaSaturn* y Nintendo64, desequilibró definitivamente esta situación, pues las nuevas máquinas domésticas igualaban e incluso superaban tecnológicamente a la mayoría de recreativas. Por otro lado el rápido crecimiento del sector de la telefonía móvil, con aparatos cada vez más potentes que permitían ejecutar videojuegos de alta calidad e incluso emular antiguas videoconsola, habían marcado una revolución obligando a los dueños de los salones recreativos a replantearse su modelo de negocio.

Mientras tanto, el mercado de los PC seguía dominando con los típicos juegos que ya lo habían hecho con anterioridad. Triunfaban los juegos de estrategia y los *shooters*. En 2004 sale al mercado la nueva Nintendo DS, primer producto que intenta desmarcarse con la pantalla táctil y poco después sony PSP, una consola portátil de alta potencia que no consiguió alcanzar a la NDS en ventas. en 2005 Microsoft lanza su Xbox 360 y sony responde con Playstation 3, una consola con altos precios de producción y esto les provocó menos éxito del esperado. Por último en el año 2013 sony saca a la venta su Playstation4 junto con microsoft que lanza la Xbox One, a la espera de muchos éxitos por sus bajos costes, aunque aún un corto mercado de videojuegos.

Figura 14. Xbox One (izquierda) y Playstation4 (derecha)

2.5. Game Engines en la actualidad

2.5.1. Unity3D

A diferencia del motor desarrollado para este proyecto Unity3D, va dirigido a juegos de alto rendimiento, como podrían ser los juegos actualmente conocidos en el mercado como triple A, dentro de este motor es posible programar un comportamiento basado en tiles, como el utilizado en el proyecto, pero no es algo para lo que esté adaptado y supondría un esfuerzo mayor. En unity3D es necesario tener una aplicación de escritorio para desarrollar juegos, también para jugarlos, aunque en navegador solo necesitaremos un plugin(Unity Web Player).

Unity es un motor de videojuego multiplataforma creado por Unity Technologies.

Unity está disponible como plataforma de desarrollo para Windows y OS X, y permite crear juegos para Windows, OS X, Linux, Xbox 360, PlayStation 3, Wii, Wii U, iPad, iPhone, Android y Windows Phone. Gracias al Plug-In Web de Unity, también se pueden desarrollar juegos de navegador, para Windows y Mac. Su última versión, la 4.3, fue lanzada el 12 de noviembre de 2013. Desde la página oficial se pueden descargar dos versiones: Unity y Unity Pro.

El éxito de Unity ha llegado en parte debido al enfoque en las necesidades de los desarrolladores independientes que no pueden crear ni su propio motor del juego ni las herramientas necesarias o adquirir licencias para utilizar plenamente las opciones que aparecen disponibles. El enfoque de la compañía es "democratizar el desarrollo de juegos", y hacer el desarrollo de contenidos interactivos en 2D y 3D lo más accesible posible a tantas personas en todo el mundo como sea posible.

En 2008, con el auge del iPhone, Unity fue uno de los primeros desarrolladores de motores en empezar a apoyar a la plataforma en su totalidad. Unity está siendo utilizado por el 53,1% de los desarrolladores con cientos de juegos lanzados en dispositivos Android y iOS.

En 2009, Unity comenzó a ofrecer una versión de su producto de forma gratuita. El número de desarrolladores registrados empezó a crecer rápidamente tras el anuncio. En abril de 2012, Unity se dio cuenta del nuevo nivel popularidad a medida que el recuento de los desarrolladores registrados alcanzaron un millón, 300.000 de los cuales utilizan Unity sobre una base mensual regular.

La primera versión de Unity se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido solamente para funcionar y generar proyectos en los equipos de la plataforma Mac y obtuvo el éxito suficiente como para continuar con el desarrollo del motor y herramientas. Unity 3 fue lanzado en septiembre de 2010 y se centró en empezando a introducir más herramientas que los estudios de alta gama por lo general tienen a su disposición, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas que normalmente serían difíciles de conseguir en un paquete asequible. La última versión de Unity, Unity 4, lanzada a finales de 2012, se anunció en junio de 2012 e incluye añadidos como Mecanim animation, soporte para DirectX 11 y soporte para juegos en Linux.

Unity soporta la integración con 3ds Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks y Allegorithmic Substance. Los cambios realizados a los objetos creados con estos productos se actualizan automáticamente en todas las instancias de ese objeto durante todo el proyecto sin necesidad de volver a importar manualmente.

El motor gráfico utiliza Direct3D (Windows), OpenGL (Mac, Linux), OpenGL ES (Android, iOS), y APIs propietaris(Wii). Tiene soporte para mapeado de relieve, reflexión de mapeado, mapeado por paralaje, pantalla de espacio oclusión ambiental (SSAO), sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa.

Se usa ShaderLab language para el uso de shaders, soporta tanto programación declarativa de los programas de función fija de tuberías y shader GLSL o escritas en Cg. Un shader puede incluir múltiples variantes y una especificación declarativa de reserva, lo que permite a Unity detectar la mejor variante para la tarjeta de vídeo actual y si no son compatibles, recurrir a un shader alternativo que puede sacrificar características para una mayor compatibilidad.

El soporte integrado para Nvidia (antes Ageia), el motor de física PhysX, (a partir de Unity 3.0) con soporte en tiempo real para mallas arbitrarias y sin piel, ray casts gruesos, y las capas de colisión.

El scripting viene a través de Mono. El script se basa en Mono, la implementación de código abierto de .NET Framework. Los programadores pueden utilizar UnityScript (un lenguaje personalizado inspirado en la sintaxis ECMAScript), C# o Boo (que tiene una sintaxis inspirada en Python). A partir de la versión 3.0 añade una versión personalizada de MonoDevelop para la depuración de scripts.

Unity también incluye Unity Asset Server - una solución de control de versiones para todos los assets de juego y scripts, utilizando PostgreSQL como backend, un sistema de audio construido con la biblioteca FMOD, con capacidad para reproducir audio comprimido Ogg Vorbis, reproducción de vídeo con códec Theora, un motor de terreno y vegetación , con árboles con soporte de billboarding, determinación de cara oculta con Umbra, una función de iluminación lightmapping y global con Beast, redes multijugador RakNet y una función de búsqueda de caminos en mallas de navegación.

Figura 15. Logo de Unity3D

2.5.2. Unreal Engine

De la misma forma que Unity3D, va dirigido a juegos de alto rendimiento, para videoconsolas de última generación, dentro de este motor es posible programar un comportamiento basado en tiles, como el utilizado en el proyecto, pero no es algo para lo que esté adaptado y supondría un esfuerzo mayor. Unreal Engine, a medida que acaba la generación para la que desarrolla videojuegos, va creando versiones públicas de este motor para todo tipo de desarrolladores.

Unreal Engine es un motor de juego de PC y consolas creados por la compañía Epic Games. Implementado inicialmente en el shooter en primera persona. Unreal en 1998, siendo la base de juegos como Unreal Tournament, Deus Ex, Turok, Tom Clancy's Rainbow Six: Vegas, America's Army, Red Steel, Gears of War, BioShock, BioShock 2,BioShock Infinite Star Wars Republic Commando, Batman: Arkham Asylum o Mass Effect. También se ha utilizado en otros géneros como el rol y juegos de perspectiva en tercera persona. Está escrito en C++, siendo compatible con varias plataformas como PC (Microsoft Windows,GNU/Linux), Apple Macintosh (Mac OS, Mac OS X) y la mayoría de consolas (Dreamcast, Gamecube, Wii, Xbox, Xbox 360, PlayStation 2, PlayStation 3). Unreal Engine también ofrece varias herramientas adicionales de gran ayuda para diseñadores y artistas.

La última versión de este motor es el Unreal Engine 3, está diseñado para la tecnología:

- DirectX 9 (para las plataformas Windows XP/Vista/7 de 32/64-bit y Xbox 360).
- DirectX 10 (para plataformas Windows Vista/7 32/64-bit).
- OpenGL (para plataformas Linux, Mac OS X de 32/64-bit y PlayStation 3).

El Unreal Engine de tercera generación aparece en 2006, diseñado para PC con soporte DirectX 9/10, Xbox 360 y PlayStation 3. Su motor reescrito soporta técnicas avanzadas como HDRR, normal mapping, y sombras dinámicas. Incluyendo componentes para herramientas complementarias al igual que las anteriores versiones del motor. Se sustituye a Karma por PhysX de Ageia (posteriormente adquirido por NVIDIA), y FaceFX se incluye además para generar animaciones faciales. Epic utilizó esta versión del motor para el videojuego Gears of War y Unreal Tournament 3, posteriormente utilizando una versión mejorada para Gears of War 2.

En el E3 de 2007 Sony anuncia que se asocia a Epic para optimizar el motor para el hardware del PlayStation 3, cambios que se utilizan actualmente por los desarrolladores de juegos de esta

plataforma. Asimismo surge el anuncio del desarrollo de una modificación del Unreal Engine para la Wii.

Debido a su política de licencias, Epic obtuvo numerosos contratos con compañías como Atari, Activision, Capcom, Disney, Konami, Koei, 2K Games, Midway,THQ, Ubisoft, Sega, Sony, Electronic Arts, Square Enix, CCP Games y 3D Realms, entre otras.

En la GDC de 2008, Epic revela numerosas mejoras de diseño, entre las que se incluyen: renderizado para mayor número de objetos simultáneos, físicas más realistas para efectos de agua, físicas de texturas corporales, mayor destructibilidad para los entornos, IA mejorada y efectos mejorados en luces y sombras con rutinas avanzadas para los shaders. Esta revisión (conocida como Unreal Engine 3.25/5) hizo su debut con el título Gears of War 2.

El Unreal Engine 3 además se aplica en sectores no relacionados con los videojuegos como simulación de construcciones, simuladores de conducción, previsualización de películas y generación de terrenos.

El 5 de noviembre del 2009 Epic Games publicó una versión gratuita del Unreal Development Kit para permitir a grupos de desarrolladores amateur realizar juegos con el Unreal Engine 3.

Figura 16. Logo Unreal Engine (izquierda) y Logo Epic Games (derecha)

2.5.3. RPG maker

Este motor es el más parecido al desarrollado en el proyecto que podemos encontrarnos actualmente en el mercado, también se basa en tiles, pero está limitado a hacer juegos del género RPG y por turnos(TBS), si nos queremos desviar de este género cuando desarrollamos un videojuego, el esfuerzo equivale a hacerlo de cero, pero es posible. Para utilizar este programa es necesaria una aplicación de escritorio y también genera aplicación de escritorio.

El nombre japonés, Tsukūru es un juego de palabras mezclando la palabra japonesa tsukuru (作る), que significa "hacer" o "crear", con tsūru (ツール), la transcripción japonesa de la palabra inglesa "tool" (herramienta, utilidad).

La serie RPG Maker fue publicada inicialmente en Japón, posteriormente con versiones en Hong Kong, Taiwan, y los Estados Unidos.

Los RPG Maker permiten al usuario crear sus propios videojuegos de rol. Incluyen un editor de mapas, un editor de eventos y un editor de combates.

Todas las versiones de RPG Maker de PC tienen el RTP (Run Time Package) que incluye materiales como gráficos para mapeados, personajes, música, efectos de sonido, etc. que pueden ser utilizados para crear nuevos juegos.

Una característica interesante de las versiones de PC de RPG Maker es que el usuario puede agregar nuevos materiales gráficos y sonoros personalizados al proyecto.

Muchos sitios y comunidades en la red se dedican a ayudar y compartir sus creaciones con los usuarios, también se dedican a compartir gráficos, sonidos, fondos y demás archivos que ayudan a la elaboración de una nueva creación, o la modificación de una ya creada anteriormente.

Curiosamente las versiones para PC poseen una base de datos en la cual la característica, de eventos comunes que permite desarrollar (que son los equivalentes a Scripts en lenguajes de programación) al usuario elaborar prácticamente cualquier tipo de juego, incluso tipo plataforma como Mario Bros o ARPG (Action RPG, o rol de acción) como La leyenda de Zelda.

RPG Maker 95 (popularmente abreviado como RM95 o RPG95) es la primera versión publicada para Microsoft Windows, en japonés, por parte de ASCII. A pesar de ser una versión más antigua,

RPG Maker 95 dispone de mayor resolución de pantalla y de gráficos de mapa y personajes que su sucesor, RPG Maker 2000.

RPG Maker 2000 (popularmente abreviado como RM2k) es la segunda versión de RPG Maker para Microsoft Windows. A pesar de la reducción de resolución gráfica con respecto a RPG Maker 95, RPG Maker 2000 dispone de más prestaciones, por ejemplo, los gráficos de mapeados y personajes se cargan en archivos por separado, a diferencia de su predecesor, el cual quedaba muy limitado.

RPG Maker XP (popularmente abreviado como RMXP) es el primer RPG Maker que hace uso de Ruby, proporcionándole capacidades de programación. Por otra parte, numerosas características bastante habituales y utilizadas en RPG Maker 2000 y 2003 se le han eliminado.

RPG Maker XP trabaja a igual resolución gráfica que RPG Maker 95. Destaca el tamaño gráficos de personaje, la mayor profundidad de color y transparencias parciales. También existen otros cambios a nivel técnico, como mayor número de capas de edición en los mapas, en concreto tres. Incluye herramientas como un gestor de archivos que contiene desde tilesets hasta música para el juego desde o hacia su ordenador.

RPG Maker XP posee comandos de evento con los que es posible realizar una gran variedad de operaciones, ya sea con matemáticas como las variables, u otros procesos para el juego como cambiar personajes, etc. Para principiantes es más recomendable utilizar el RPG Maker VX o 2003, ya que tienen más funciones que se perdieron en el RMXP, las cuales necesitan ser programadas en los Scripts en esta versión: los Scripts se hacen de forma fácil para un programador (Ruby es de programación muy sencilla), pero de forma difícil para alguien que no esté familiarizado con la programación.

Con los scripts puedes hacer juegos no necesariamente de RPG. Incluso es posible la creación de cualquier tipo de programa, por ejemplo un programa que imita a la aplicación RMXP pero con más funciones (se han realizado, pero tienen el inconveniente que exigen muchos recursos en comparación a la aplicación original) o de cálculos para encontrar nuevos números perfectos.

Figura 17. Logos de RPG Maker

2.5.4. Adventure Game Studio

Adventure Game Studio, o AGS, es un *game engine* de software libre que se distribuye gratuitamente y se centra en la creación y desarrollo de aventuras gráficas, dispone de varias herramientas para facilitar la edición del videojuego y además incluye un entorno de desarrollo integrado que permite manipular todos los aspectos del juego mediante scripts basados en C.

La primera versión de AGS fue creada por Chris Jones y lanzada en 1997. Esta primera versión sólo permitía la creación de videojuegos con gráficos de baja resolución y control con teclado.

Durante los primeros años tan sólo se desarrollaron pequeñas demos que no hacían más que testear las posibilidades del *game engine*, no fue hasta el año 2000 cuando, con el lanzamiento de Larry Vales y Rob Blanc el *game engine* comenzó a hacerse popular. Actualmente cuenta con una activa comunidad que desarrolla aventuras gráficas, e incluso juegos de otros géneros, con AGS.

En enero de 2008 el editor fue reescrito totalmente usando .NET Framework soporte para aceleración 3D.

En octubre de 2010 se publicó el código fuente de AGS bajo Licencia Artística v2.

Algunos de los juegos desarrollados con AGS han sido *Gemini Rue*, que ganó varios premios en 2011 entre ellos *Adventure Gamers' Best Story & Best Independent Adventure Game of the Year*, o *The Journey Down*, cuya buena recepción hizo que los desarrolladores se decidieran a relanzar el título con su propio *game engine*.

Figura 18. Logo de AGS

3. Géneros y subgéneros

A continuación sugerimos una lista de géneros y subgéneros de videojuegos que han ido apareciendo durante la historia. De los cuales la mayoría podrían crearse con nuestro motor siempre y cuando lo transformemos en el formato de *tiles.*

3.1. Acción

Los juegos de acción son aquellos en los que se destaca la habilidad y reflejos del jugador. Este género es el más amplio del mundo de los videojuegos, englobando muchos subgéneros como los explicados a continuación. La mayoría de los videojuegos que alcanzan ventas desorbitadas están incluidos en este género.

Si bien los objetivos de estos videojuegos varían de videojuego a videojuego, generalmente implican avanzar a través de niveles, eliminando hordas de enemigos y resolviendo problemas. Muchos videojuegos incluyen uno o más "jefes", a veces precedidos por "minijefes". Un minijefe es generalmente el clímax hacia un nivel o serie de niveles, con un jefe al final del juego o periódicamente por el juego, llevando a un "jefe final", el cual derrotar es el objetivo principal. También un "jefe" en las fases iniciales puede transformarse posteriormente en "minijefe". Para derrotar jefes se suele usar el "reconocimiento de patrones" y la velocidad de reacción física. En la mayor parte de los videojuegos viejos (e inclusive algunos modernos) los jefes son programados con un patrón de ataques simple o con movimientos que el jugador aprende a través de la experiencia. Estos patrones simples a menudo abarcan "combos" que exigen al jugador saltar, esquivar o bloquear ataques para luego atacar en ciertos puntos claves, todo esto mientras se maneja el tiempo de los patrones para poder atacar.

Muchos subgéneros, como los videojuegos de plataformas, incluyen problemas de estilo gimnástico, como por ejemplo saltos de tiempos regulados hacia y desde plataformas móviles. Los videojuegos de plataformas, ya sean en dos o tres dimensiones, son similares en concepto a la serie de videojuegos Mario Bros.. Algunos videojuegos de acción tienen una jugabilidad al estilo de los videojuegos de disparos en tercera persona, permitiéndole al jugador adquirir y (a veces) mejorar un conjunto de armas, cada una con una habilidad especial.

Otro subgénero común es el shoot 'em up, en el cual el jugador controla un personaje o vehículo con muchas armas, y debe disparar a una gran cantidad de enemigos y objetos.

3.1.1. Shooter

Los videojuegos de disparos o *shooters* son un subgénero que se podría incluir dentro de muchos otros géneros, ya que existen shooters de todo tipo. Pero también comparten la característica de que por norma se controla a un personaje que acostumbra a controlar un arma que puede ser disparada.

Características:
Los shooters suelen compartir perspectivas parecidas, como por ejemplo una visión en primera persona, que son conocidos como los *first person shooter*(FPS), o con la cámara al hombro del personaje principal en tercera persona, o con una vista lateral en la que vamos avanzando y eliminando enemigos, estos últimos se consideran *shooters arcade*.

Actualmente, una de las características más importantes de un shooter, es que sea *online* multijugador, para aumentar el trabajo en equipo y la diversión.

Subgéneros dentro de los shooters:

- **Shoot'em up:** Los *shoot'em up* se suelen considerar juegos arcade en los que incluimos juegos como *Metal Slug*. Acostumbran a ir avanzando en un escenario con scroll en el que van apareciendo enemigos y hay que ir derrotándolos con armas de fuego.

Figura 19. Metal Slug para recreativa

- **Third person shooter:** En este género el personaje es visto desde la parte de atrás y con un cursor con forma de punto de mira que nos permite apuntar, el personaje mirará hacia donde estemos apuntando y ahi es donde disparará. Muchos juegos actualmente se están adaptando a esta nueva perspectiva como, Gears of war, Max Payne, Splinter Cell, Uncharted, Socom.

Figura 20. Socom 2: US Navy Seals para PS2

- **Shooter sobre railes:** En estos tipos de *shooter* el jugador no tiene control sobre el movimiento del personaje principal, solo sobre su puntería. La cámara se va moviendo sobre un raíl y el jugador se dedica a eliminar a los enemigos. En este subgénero podemos incluir juegos como Kid Icarus, Starfox o Panzer Dragoon.

Figura 21. Kid Icarus Uprising para N3DS

- **Galería de tiro:** Acostumbran a ser jugados con una pistola como periférico, son aquellos en los que el juego nos construye un escenario y disparamos a todos los enemigos, una vez superada esta fase, el juego avanza hacia otra escena y así hasta terminar con las monedas introducidas en la máquina o improbablemente el juego. Este subgénero incluye juegos como *time crisis* o *House of the dead.*

Figura 22. Time Crisis para recreativa

- **Shooters híbridos:** Son aquellos que combinan varios elementos de otros géneros como puede ser plataformas, estrategia o rol. En este subgénero se incluyen juegos como Fallout, Deus Ex, Oblivion o Mirror's Edge.

Figura 23. Mirror's Edge para PS3

- **First person shooter:** Los first person shooter, son aquellos que se basan en disparar desde la visión del personaje que sostiene el arma, intentan mostrar todo lo que él ve y siente. En este subgénero se pueden incluir juegos como Call of duty, Golden eye, Perfect dark, Turok o Counter Strike.

Figura 24. Goldeneye para Nintendo 64

3.1.2. Arcade

Arcade es el término que se utiliza para las máquinas recreativas. Este género engloba principalmente a las máquinas antiguas como el pinball o las máquinas tragaperras de los casinos. Pero actualmente se puede considerar un juego arcade todo aquel que proviene o es similar a los juegos de las máquinas recreativas.

Características:
Las máquinas son muebles dotados de controles, de aquí proviene el nombre de consola en lo que evoluciona a videoconsola. Para poder jugar una partida en estas máquinas hay que introducir una moneda, habitualmente una moneda equivale a una vida en el juego, hay diversos métodos de pago dependen del salón en el que se encuentre la máquina y de la misma máquina, como tarjetas fichas o carnets.

La característica común de las máquinas recreativas es la escasa duración de las partidas, porque se acabe el tiempo o las vidas también conocidas como créditos. Así la máquina asegura que el jugador siga introduciendo monedas. Intentan evitar la complejidad del juego para conseguir enganchar a cualquier tipo de persona.

Unas de las características más buscadas entre los usuarios de estas máquinas, es tener un gran valor de reusabilidad. Así cuando el jugador ha terminado el juego satisfactoriamente, hay un deseo de comenzar de nuevo inmediatamente o en una ocasión próxima. Con este objetivo, gran parte de las máquinas de *arcade* poseen sistemas de puntuación que miden los logros del jugador, existiendo la opción de jugar por terminar el juego y/o competir contra otros jugadores que obtuvieron mejor puntuación.

En otros términos se conoce como juego arcade a aquel juego que recuerda a los juegos que se jugaban en las máquinas recreativas.

En este género se pueden incluir todo tipo de juegos, los más conocidos son los shooters de galería o los *beat'em up* en los que una vida equivale a una moneda.

Figura 25. Final Fight arcade (izquierda) y Captain Tsubasa(derecha)

3.1.3. Beat'em up

El *beat'em up* es un género de videojuegos en el que se destaca el combate cuerpo a cuerpo entre el personaje controlado por el jugador y un gran número de enemigos que quieren acabar con el. Los personajes pueden portar todo tipo de armas aunque no suelen predominar las armas de fuego.

Características:

Los *beat'em ups* es un género distinto al género común de lucha. Los *beat'em ups* ocurren sobre un nivel largo, con *scroll* lateral de la pantalla cuando el jugador se mueve sobre el escenario. Los juegos de enfrentamientos competitivos han evolucionado para incluir mayor variedad de ataques

que el jugador puede usar, mientras que los *beat'em ups* ofrecen un esquema de control más simple ya que están más orientados a las máquinas recreativas.

En este tipo de juegos, uno o más jugadores cada uno elige un único personaje y forma un equipo para dar golpes en su camino a una multitud de hordas de enemigos controlados con una inteligencia artificial bastante simple. Acostumbran a contener un poderoso "jefe" final. En este subgénero pueden incluirse juegos como *Double Dragon* o *Tenage Mutant Ninja Turtles.*

Figura 26. Double dragon para Megadrive (izquierda) y Teenage Mutant Ninja Turtles para SNES (derecha)

3.1.4. Hack & Slash

Este género traducido como "corta y raja", se basa principalmente en combates frenéticos entre un jugador principal y un conjunto de hordas seguidas de un "jefe" final. El estilo es muy parecido a los *beat'em up*, a diferencia de que el personaje principal suele llevar un arma propia y suelen desplazarse en entornos 3D en los que pueden atacar en todas las direcciones y no solo hacia uno o dos lados.

Características:

Los *hack & slash* suelen combinarse con otros géneros como el rol, la aventura o las plataformas. Se basan en un conjunto de hordas que te intentan impedir el paso hacia un "jefe" final, el cual te desbloqueará una nueva habilidad, arma o proporcionará una cantidad de trofeos suficientes para poder completar el siguiente nivel.

En este subgénero se pueden incluir juegos como: Gauntlet, Devil my cry, God of war, Ninja Gaiden, Heavently Sword, lollipop Chainsaw, Dante's inferno, Bayonetta, MadWorld, Brutal Legend, Castlevania, Metal Gear Rising o No More Heroes. Este es uno de los subgéneros modernos con un mercado de videojuegos más amplios.

Figura 27. God of war 3 para PS3 (izquierda) y Lollypop chainsaw para XBOX 360 (derecha)

3.1.4. Plataformas

Los videojuegos de plataformas son un género que se caracteriza por tener que caminar, correr, saltar o escalar sobre una serie de plataformas y acantilados, con enemigos sencillos, mientras se recogen objetos para poder completar el juego. Este tipo de videojuegos suelen usar vistas de desplazamiento horizontal hacia la izquierda o la derecha. Es uno de los géneros más populares desde la creación de Mario Bross.

Características:

Acostumbran a moverse en un *scroll* lateral mientras el personaje principal intenta llegar al final de la pantalla recolectando el máximo de trofeos posibles. Mario bross inició este género dándole un giro al género cuando el mismo introduce el subgénero plataformas 3D, con Mario 64 para Nintendo 64. Incluye las mismas características, correr hasta llegar al objetivo saltar, intentar evitar enemigos o eliminarlos, excepto el scroll lateral en 2D, que acostumbra a ser visión en tercera persona con la cámara detrás del personaje protagonista.

Este subgénero incluye juegos como Mario bross y todos los de su saga, Sonic, ICO, Shadow of the colossus, Peratallada, Crusafont, 1714 the game, Mirror's Edge o guacamelee.

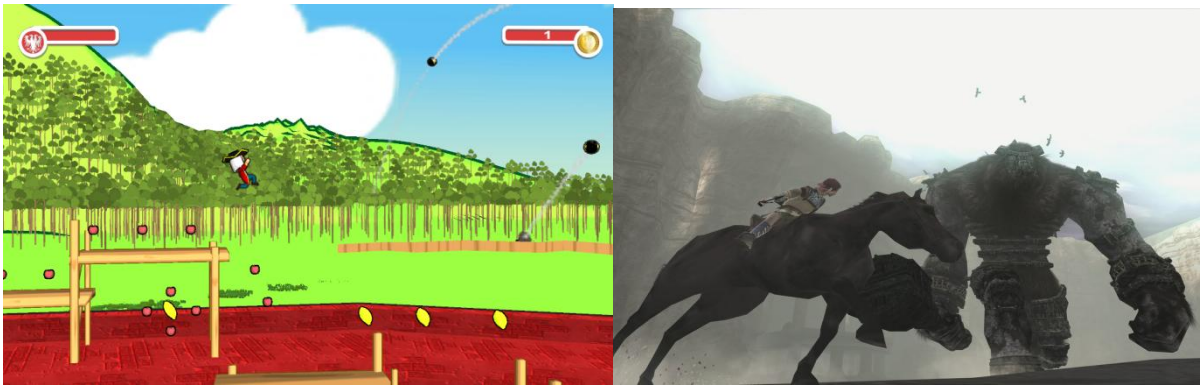


Figura 28. 1714 the game, para pc (izquierda), shadow of the colossus para PS2 (derecha)

3.2. Aventuras

El género de aventuras fue uno de los primeros que surgieron durante las primeras de la historia de los videojuegos. En este tipo de videojuegos, el jugador encarna al protagonista de una historia interactiva en la que debe explorar el entorno y resolver puzzles para avanzar. Éste género da más énfasis a la narrativa que a la jugabilidad y generalmente están diseñados para ser jugados en solitario.

El término "juego de aventuras" viene del primer juego de este tipo que se desarrolló, *Colossal Cave Adventure*, o *Adventure,* como se le conoció en 1970, fecha de su creación. Fue desarrollado por Will Crowther como un entretenimiento para sus hijas, pero el juego se hizo muy popular en ARPAnet y fue ampliado por los usuarios de la red.

Los elementos típicos de éste género son:

- Los puzzles: el jugador avanza en la historia resolviendo los puzzles que se le van presentando y cada vez que resuelve uno se le recompensa con un nuevo fragmento de la historia que se desarrolla a su alrededor.

- Recolección y uso de objetos: típicamente, el jugador irá recogiendo objetos que encuentre durante su aventura y los utilizará de forma ingeniosa para resolver los puzzles que se le presenten.

- Desarrollo narrativo: debido a que la aventura se centra más en la narrativa, el personaje se desarrolla de forma parecida a como lo hacen en la literatura y el cine, aprendiendo y madurando gracias a las experiencias que viven, en lugar de hacerse más poderosos como suele pasar en otros géneros.

- Diálogos: en la gran mayoría de videojuegos de aventuras, el protagonista interactuará con otros personajes, esto suele dar lugar a conversaciones y a "árboles conversacionales" en los que el jugador puede decidir qué dirá el personaje y de esto dependerá la respuesta del personaje con el que habla.

3.2.1. Aventura conversacional

Figura 29. Mystery, la primera aventura conversacional con gráficos y el primer juego que hicieron los fundadores de Sierra Online

Las aventuras conversacionales fueron las primeras que surgieron y de los primeros videojuegos que se comercializaron, estas aventuras se caracterizaron porque toda la narrativa se representa mediante text y el jugador interactuaba escribiendo la acción que quería realizar, por esta razón el juego debía de disponer de un parser que reconociera los textos que pudiera escribir el jugador.

Éste subgénero se inauguró con el ya citado *Colossal Cave Adventure*, al principio carecían totalmente de gráficos y todo el escenario era descrito por el texto, esto fue hasta *Mystery House*, desarrollado por Ken y Roberta Williams, que fue la primera aventura con gráficos, este juego se diferenciaba tan solo por el hecho en que el escenario estaba descrito por un dibujo en lugar de por texto. Este juego fue el primero de muchos más que desarrollaron esta pareja y les llevó a fundar *Sierra Entertainment*.

Este género fue líder de ventas durante la primera mitad de la década de los 80, momento en que fue desbancada por las aventuras gráficas.

3.2.2. Aventura gráfica

La aventura gráfica, o también llamadas aventuras *point n' click*, fue el sucesor de la aventura conversacional, la estructura de juego era la misma, pero la aventura gráfica contaba con gráficos animados y se podía mover al protagonista por la pantalla. El primer juego de este tipo fue *King's Quest I: Quest for the Crown*, desarrollado por *Sierra Entertainment*, las interacciones con el entorno todavía debían escribirse con el teclado, pero sentó las bases de un subgénero que fue líder del mercado de videojuegos para PC hasta bien entrada la década de los noventa.

Visto el éxito de estas aventuras, LucasFilms creó una división de videojuegos, llamada LucasFilms Games que se dedicó al desarrollo de aventuras gráficas. Fue esta compañía la que desarrolló *Maniac Mansion*, que utilizaba el Game Engine SCUMM y fue la primera en dejar de lado el parser de verbos e integró el uso del ratón para controlar al personaje.

Las aventuras gráficas fueron todo un éxito hasta mediados de los noventa, en que juegos de acción más dinámicos, como los FPS, desbancaron a las aventuras gráficas, dejando a éste género casi abandonado hasta la actualidad, en que gracias a nuevos modelos de financiación como el crowdfunding y al auge de las desarrolladoras independientes está aumentando de nuevo el número de aventuras gráficas desarrolladas.

Figura 30. King's Quest I (izquierda) y Maniac Mansion (derecha)

3.2.3. Película interactiva

La película interactiva es un subgénero a caballo entre el cine y el videojuego, este tipo de aventuras nunca ha sido muy común, pero han ido saliendo de forma esporádica desde el principio del género, e incluso hay alguna compañía que se ha especializado en este tipo de videojuegos, como Quantic Dream.

Estas aventuras se caracterizan por narrar una historia en la que el jugador solo tiene interacciones limitadas con ésta, pero que pueden cambiar de forma notable el transcurso de la historia, algunas pueden añadir algún minijuego para dar algo más de jugabilidad, pero estos minijuegos raramente aportan nada al desarrollo del juego.

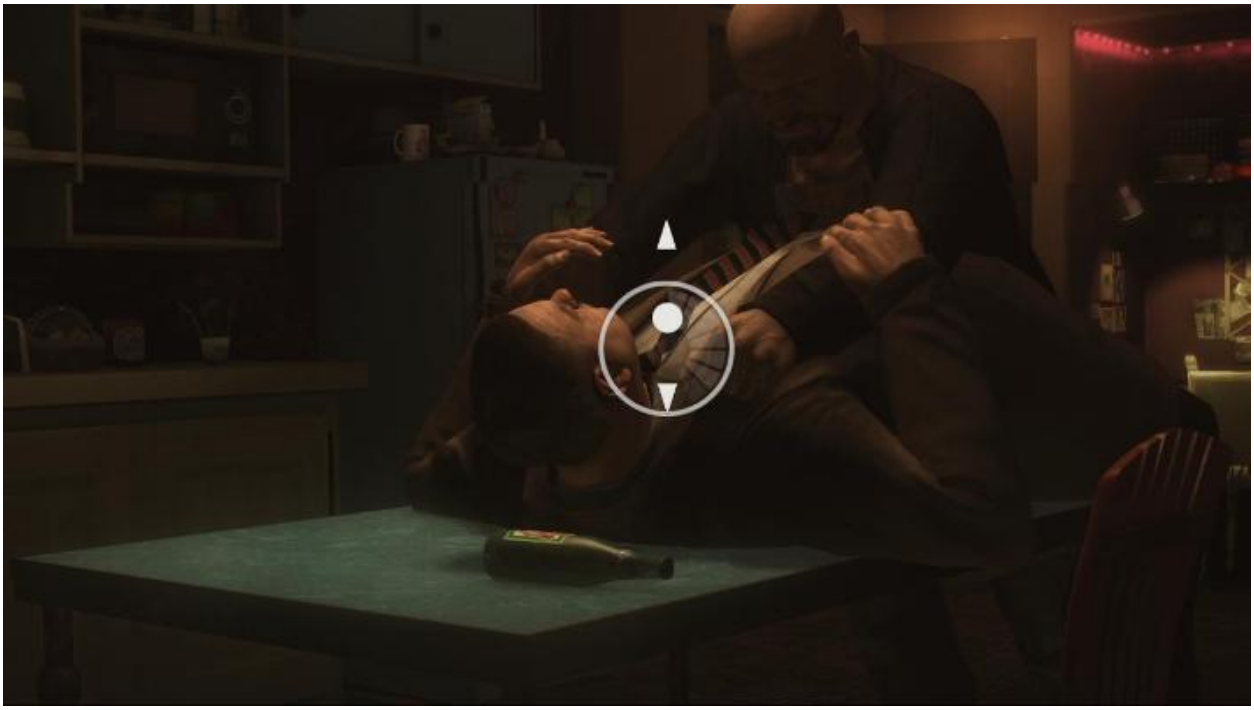


Figura 31. Heavy Rain, película interactiva desarrollada por Quantic Dream

3.3. RPG

El género de los *Role Playing Games* (RPG) tienen sus orígenes en los juegos de rol de papel tradicionales, como *Dungeons & Dragons*, y comparte con estos su complejidad, las habilidades y debilidades de los personajes están definidos por reglas que el jugador debe entender.

Los primeros videojuegos RPG eran simples aventuras donde un protagonista debía atravesar una mazmorra venciendo los enemigos que salieran a su paso, según fuera venciendo enemigos, el protagonista irá ganando experiencia y equipo que le haría haciendo más fuerte y le ayudaría a vencer los enemigos que se encontraría más adelante.

Con el tiempo, los RPG se han ido haciendo más complejos y han surgido varios subgéneros, pero todos ellos tienen en común ciertas características:

- Narrativa y escenario: Todos los RPG hacen énfasis en la historia, aunque no se centra tanto en ella como pueden hacerlo las juegos del género de aventuras, los personajes

crecen de forma parecida a ellas. Estas aventuras también suelen estar situadas en mundos de fantasía, con elementos medievales o futuristas.

- Desarrollo de personajes: Un punto en común de todos los RPG es el desarrollo del personaje, según el personaje va superando los obstáculos que se le presentan, este se va haciendo más fuerte y va desarrollando nuevas habilidades, por lo general, el jugador tiene control sobre qué habilidades desarrolla y hasta qué punto lo hace, permitiendo de esta forma cierto grado de personalización del personaje.

- Exploración y búsquedas: Un importante elemento de los RPG es la exploración, durante el juego los protagonistas viajan por el mundo y durante estos viajes conocen NPCs que les encargarán búsquedas, estas búsquedas, también llamadas *quests*, serán la excusa que tendrá el personaje para adentrarse en mazmorras y luchar contra enemigos.

- Combate: La forma que tienen los jugadores de enfrentarse a los enemigos es mediante combates, la forma en que se desarrollan estos combates depende del subgénero al que pertenezca el videojuego, pero en estos combates los protagonistas utilizarán las habilidades aprendidas durante la aventura para hacer frente a sus enemigos.

- Interfaz: Debido a la gran complejidad de este tipo de videojuegos, se suele contar con un sistema de menús que permiten al jugador visualizar y manejar toda la información sobre los personajes y sus habilidades y equipo.

Cabe destacar que, a parte de los subgéneros, hay dos "corrientes" distintas a la hora de hacer RPG, el occidental y el oriental. Los RPG occidentales se centran más en el desarrollo e interpretación del personaje, dando más libertad al jugador para escoger la forma de interactuar con el mundo, los RPG orientales, en cambio, suelen dejar muy restringida la forma de interactuar con el mundo, pero son más innovadores a la hora de crear sistemas de crecimiento para los personajes y dan más control sobre los personajes secundarios.

A parte de ésta diferenciación de estilo, los RPG también pueden dividirse en distintos subgéneros que se suelen diferenciar, generalmente, en el estilo en que se desarrollan los combates.

3.3.1. RPG Clásico

Éste tipo de RPG fueron los primeros que se hicieron, se diferencian del resto por el desarrollo de los combates. Estos combates suelen contar con un aspecto y manejo diferente al que se tiene durante las fases de exploración, muy a menudo los combates son aleatorios y aparecen de repente mientras el protagonista se mueve por la mazmorra. En estos combates, los protagonistas y los enemigos se sitúan en extremos opuestos del escenario y esperan su turno para lanzar sus ataques al bando rival. Durante el turno de alguno de los protagonistas, el jugador escoge, mediante un menú, el tipo de habilidad que quiere utilizar y sobre quien quiere hacerlo. Generalmente estos combates acaban cuando todos los miembros de uno de los bandos quedan incapacitados.

Figura 32. Final Fantasy VII (izquierda) y Dragon Quest VIII (derecha), ejemplos de RPGs clásicos

3.3.2. Action RPG

Los Action RPG se diferencian del resto porque los combates son en tiempo real, a diferencia del resto de RPG, el estilo de representación del juego no cambia cuando empiezan los combates y el jugador puede mover al protagonista por el escenario de la misma forma como lo hace durante la fase de exploración.

En los combates de este tipo de juegos son importantes los reflejos del jugador, ya que le permiten esquivar y defenderse de ataques enemigos así como encontrar el momento idóneo para atacar. Si el protagonista tiene compañeros estos son controlados por la inteligencia artificial del juego. Muchas veces el jugador tiene la capacidad de dar directrices a los compañeros para que se comporten de cierta forma durante los combates, permitiendo preparar algunas estrategias.

Figura 33. Kingdom Hearts (izquierda) y The Elder Scrolls: Skyrim (derecha), ejemplos de Action RPG

3.3.3. Tactic RPG

Los Tactic RPG incluyen a los elementos propios de los RPG elementos de los juegos de estrategia, estos juegos se desarrollan de la misma forma como lo hace un RPG clásico en lo que respecta a la exploración y el desarrollo de personajes, pero los combates son más propios de un juego de estrategia que de un RPG.

En los combates de este tipo de juegos, dos grupos de personajes se enfrentan por turnos, en un escenario normalmente representado por un tablero. Durante el turno de cada personaje, el jugador tiene la capacidad de ordenar al personaje un número limitado de acciones, como moverse o atacar. Estos combates requieren que el jugador sepa combinar las diferentes habilidades de cada personaje así como aprovecharse de los elementos del terreno para sacar ventaja. A diferencia de lo que ocurre con el resto de RPGs, en los Tactic RPG tiene gran importancia detalles como el orden en que reaccionan los personajes, el lugar en el que se encuentran o la dirección en la que se orientan, ya que ésto puede dar ventaja a uno u otro bando. A diferencia de otros tipos de RPG, el objetivo del combate no tiene porqué ser necesariamente la eliminación del enemigo, normalmente hay más objetivos, como puede ser llegar a un punto del escenario en un número determinado de turnos o proteger a un personaje específico.

Figura 34. Final Fantasy Tactics (izquierda) y Disgaea 2 (derecha) ejemplos de Tactic RPG

3.3.4. Dungeon crawler

Los *Dungeon Crawler*, también llamados *Roguelike*, debido al primer videojuego de éste tipo, son videojuegos en los que el jugador controla a un héroe, escogido entre los disponibles al empezar, cuyo único objetivo es el de atravesar una o más mazmorras, normalmente generadas aleatoriamente, para vencer a los enemigos que en ella habitan. Pese a que estos videojuegos suelen tener una historia, el *Dungeon Crawler* no hace tanto hincapié en ella como otros RPG y se centra más en la exploración y el desarrollo de las habilidades del héroe. Este tipo de videojuegos suele permitir mucha flexibilidad en el desarrollo de los personajes, de forma que el jugador es capaz de desarrollar a un héroe totalmente personalizado.

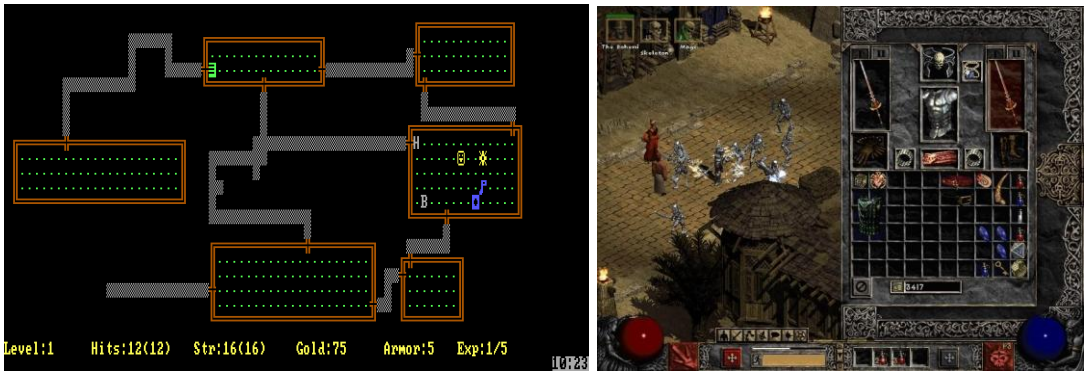


Figura 35. Rogue (izquierda) y Diablo II (derecha), ejemplos de Dungeon Crawlers

3.3.5. MMORPG

Las siglas de MMORPG vienen de *Massively Multiplayer Online Role Playing Game* y, tal como su nombre indica, son RPG online multijugador que permiten la participación de una gran cantidad de jugadores a la vez. Estos RPG destacan precisamente por eso, ya que el resto de RPG, excepto el *Dungeon Crawler*, son experiencias exclusivamente de un jugador.

En estos juegos el jugador crea un personaje, o avatar, eligiendo entre varias opciones como puede ser raza, aspecto o profesión y lo controla por un mundo poblado por otros jugadores de todo el mundo, al igual que los *Dungeon Crawler*, estos juegos están más centrados en el desarrollo del personaje y permiten una gran personalización del avatar. La narrativa de los MMORPG suele ser bastante escasa, definiendo tan solo el mundo en el que se encuentra el personaje, y dejando el desarrollo de la historia a las interactuaciones de los jugadores entre ellos.

Figura 36. World of Warcraft (izquierda) y Ragnarok Online (derecha), ejemplos de MMORPG

3.4. Simulación

Los videojuegos de simulación son aquellos que intentan recrear diferentes aspectos de la vida real. Pretenden reproducir todo tipo de sensaciones (velocidad, aceleración, percepción, dolor), para conseguir dar una experiencia real de algo que no está sucediendo en la realidad y así de esta forma no poner en riesgo la vida de alguien.

Ya habían simuladores en los años sesenta, para preparar a los pilotos de aeronaves o vehículos peligrosos a la vez que caros.

Comúnmente los simuladores requieren de periféricos concretos para cada uno. Esto aumenta el coste de su producción a la vez que reduce su expansión en el mercado. Cuanto más difícil de construir es el simulador más se acerca a la realidad y más útil es.

3.4.1. Música

El desarrollo de los simuladores musicales giran entorno a distintos tipos de juego, ya sean de tipo karaoke, de baile o en los que se tocan instrumentos musicales. La mayoria de los juegos musicales se caracterizan por necesitar un instrumento compatible con el juego como micrófonos, alfombrillas de baile, o instrumentos musicales.

Como característica principal los juegos musicales, aparte de necesitar un periférico especifico comunmente disponen de un modo multijugador para aumentar la competición.

Existen diferentes tipos de juego del estilo karaoke como *singstar, ultrastar, karaoke revolution o Lips.* En estos juegos el videojuego requiere de un micrófono con el que los jugadores deberán cantar lo más parecido a la realidad para obtener los puntos.

Por parte del estilo musical de baile hay varios juegos de simulación como *dance central*, DDR(dance dance revolution, muy conocido entre las máquinas recreativas de los salones recreativos) o *StepMania*, normalmente necesitan de un panel de baile que se coloca en el suelo sobre el que deberemos bailar colocando los pies en el momento exacto en la posición que se nos indica en pantalla.

En los últimos años, *Rock band* y *Guitar hero* revolucionaron el género simulador musical por sus grandes cantidades de ventas, ambos juegos son juegos musicales con un periférico intentando imitar la forma de una guitarra eléctrica, años atrás habían otros juegos de guitarra como *Guitar freaks*. Más adelante estos juegos evolucionaron y adaptaron sus periféricos incluso a la forma de una batería, bajo e incluso incluyeron la voz como en un juego de karaoke. También la compañía de *Guitar Hero*, fueron más allá creando DJ *hero* en el que el periférico era una mesa de mezclas que deberíamos pulsar como nos indica en pantalla. En pc surgió una imitación de *Guitar Hero* llamado *frets on fire* dado sus grandes éxitos en videoconsola.

Pero sin duda la tecnología evoluciona y en los últimos años han cambiado de tocar una guitarra de plástico con 5 botones para *guitar hero* o *Rock Band* a tocar una guitarra de verdad en *Rock Smith* en el que nos vienen la nota y el traste en el que tenemos que situar el dedo de la guitarra real que conectamos con un cable especial a nuestra videoconsola y el juego nos controla si hemos tocado la nota correcta.

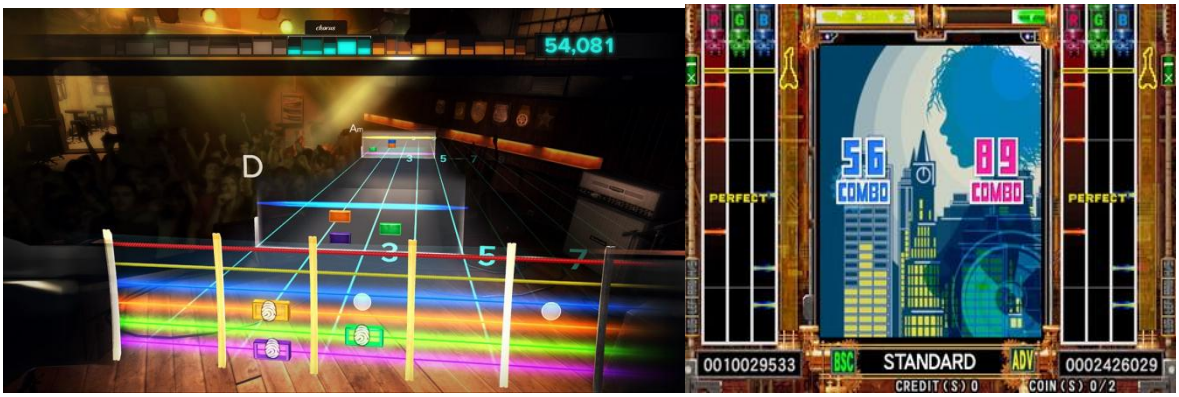


Figura 37. RockSmith para PS3 (izquierda) y Guitar Freaks para PSX (derecha)

3.4.2. Deporte

Los videojuegos que simulan el deporte son aquellos que intentan parecerse al máximo al deporte real, entre ellos encontramos, golf, tenis, fútbol, hockey, juegos olímpicos, baloncesto, etc. El jugador lo juega con un periférico común como el mando o teclado y ratón. El propósito es el mismo que en el deporte real, intenta seguir las mismas normas que en la realidad y cumplirlas rigurosamente.

Las características comunes de estos juegos, es que son juegos de gama muy alta, en los que las empresas dedican gran cantidad de dinero para su producción, suelen ser juegos que están por las zonas más altas de los *top* ventas.

Sin duda *Electronic Arts* es la compañía madre en este género ya que es la desarrolladora de juegos como la serie FIFA, la serie NBA, la serie NHL, la serie NFL, la serie Tiger Woods e incluso simuladores de combate como el Boxeo con Fight Night. Aunque en los últimos años en lo que a baloncesto se refiere NBA 2k de la compañía 2K está tomando la delantera.

Figura 38. Fifa 2014 para PS4 (izquierda) y Nba 2K14 para PS4 (derecha)

3.4.3. Vida

El simulador de vida se enfoca en controlar uno o varios personajes con capacidades humanas y emociones humanas, así controlando todos sus aspectos cotidianos, desde donde vivirá, que estudiará, hasta con quien se casará. El realismo colocado en estos juegos y una línea de tiempo que permite al personaje evolucionar, comer, dormir, convivir con otras personas, envejecer e incluso morir, hace que muchas personas tomen estos juegos como desahogo de todo lo que quisieran experimentar en su vida propia, pero sin riesgos. El más popular y culpable de definir este género es el juego de *The Sims*, también juego iniciador en introducir al público femenino en los videojuegos.

Más adelante en 2008 llegó una evolución del simulador de vida, *Spore*, un juego donde se comienza siendo una bacteria, y dependiendo de las decisiones tomadas por el jugador, la criatura puede evolucionar de diferentes formas.

Figura 39. Sims 3 para PC

3.5. Estrategia

Los videojuegos de estrategia son juegos que requieren de una planificación habilidosa de las acciones a realizar para conseguir la victoria. El jugador se enfrenta a uno o más rivales, ya sean controlados por otros jugadores o por la IA, y debe tener en cuenta varios elementos, que cambian según el juego, y utilizarlos a su favor (o en detrimento de sus rivales), para imponerse a los demás participantes, el objetivo es normalmente la eliminación de los rivales.

Los elementos de un videojuego de este tipo cambia notablemente de uno a otro, pero se pueden sacar varios aspectos comunes, que son los siguientes:

- Perspectiva y control: El escenario de juego se muestra desde una perspectiva aérea en tercera persona, desde el que podemos ver una buena parte del terreno. Estos juegos se controlan con el ratón, haciendo clic a unidades y edificios se nos permite seleccionarlos y darles órdenes.

- Unidades y conflictos: En los juegos de estrategia suelen entablarse batallas entre los jugadores. Estas batallas se llevan a cabo enfrentando a las unidades de cada bando. Cada bando tiene distintos tipos de unidades, cada uno con sus fortalezas y debilidades, y el jugador debe intentar aprovecharse de estas fortalezas para conseguir ventajas en los enfrentamientos.

- Economía y recursos: Un elemento que casi siempre aparece en los juegos de estrategia es el de la economía, ya sea de una forma más sencilla o más compleja. El jugador dispone de unos recursos, ya sean dinero, o minerales, o madera, que utiliza para financiar las construcciones y mejoras de sus infraestructuras, estas infraestructuras proveerán de mejoras que ayudarán al jugador cuando estén terminadas. Estas mejoras pueden ser de cualquier tipo, desde unidades más poderosas hasta formas de conseguir más recursos más rápidamente.

- Exploración: En la mayoría de los juegos de estrategia toda la acción se desarrolla en un terreno mayoritariamente inexplorado, es importante que el jugador explore el terreno y encuentre y ocupe el que más le favorezca y le ofrezca alguna ventaja estratégica frente a sus rivales.

Según la temática y la forma en que un juego de estrategia se desarrolla, éste puede caer dentro de uno y otro subgénero.

3.5.1. RTS

Los RTS, siglas de *Real Time Strategy* son, como su propio nombre indica juegos de estrategia en tiempo real, en este tipo de juegos, todos los participantes dan órdenes y mueven a sus unidades e infraestructuras al mismo tiempo, por lo que es importante la rapidez de decisión y ser capaz de improvisar estrategias.

Muy comúnmente este tipo de juegos están basados en escenarios bélicos, en los que se debe crear una base y un ejército que destruya las bases y ejércitos enemigos. Esto se hace acumulando recursos con los que financiar las infraestructuras de la base y el ejército, éstos recursos pueden ser cosas como alimento, madera y oro, como es el caso de *Age of Empires,* o elementos imaginarios como algunas clases de gas o cristal, elementos que se utilizan en *Starcraft*.

Figura 40. Age of Empires (izquierda) y Command & Conquer (derecha), RTS muy populares

3.5.2. TBS

Las siglas de TBS vienen de *Turn-based Strategy*, y tal como se puede suponer, son juegos de estrategia basados en turnos, a diferencia de los anteriores, en estos juegos los jugadores dan las órdenes a sus unidades en sus respectivos turnos. Normalmente no hay tiempo límite para estos turnos, por lo que el jugador tiene tiempo de sobra para pensar sus estrategias y acciones.

Normalmente estos juegos tocan un ámbito mayor que los RTS, no se centran tan sólo en las batallas y la acumulación de recursos, sino que hay más aspectos, como puede ser la cultura, el comercio o la diplomacia.

Muy habitualmente el objetivo de estos juegos son más variados y a más largo plazo que los RTS, como expandir una civilización durante miles de años y ser la más grande, o con mejor cultura.

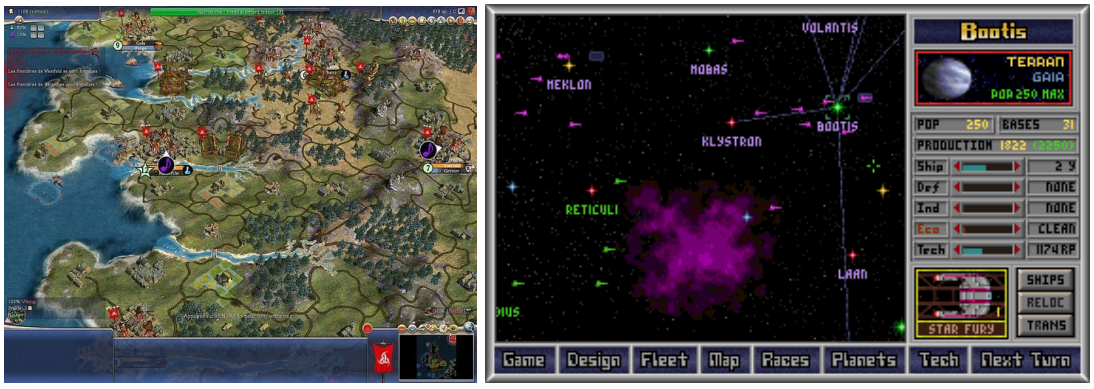


Figura 41. Civilization IV (izquierda) y Master of Orion (derecha), ejemplos de TBS

3.5.3. Negocios

Un tipo de juegos de estrategia que se diferencia de los demás por alejarse del aspecto bélico que suelen tener estos tipos de videojuegos son los juegos de simulación de negocios. En estos juegos, el jugador, en lugar de ponerse en el papel de un general o un rey, desempeña el rol de dueño de un negocio y debe administrarlo para tener éxito. Actualmente existen multitud de juegos de este tipo, en el que se manejan hospitales, parques de atracciones e incluso desarrolladoras de videojuegos.

Pese a la diferencia de enfoque de estos juegos con el resto de juegos de estrategia, estos juegos también son muy complejos y hace falta ser capaz de tener capacidad de improvisación y se han de tener en cuenta muchos aspectos para tener éxito en estos juegos.

Figura 42. Theme Park (izquierda) y Game Dev Tycoon (derecha), juegos de estrategia de negocios

3.5.4. Tower defense

En los últimos tiempos, con la popularización de los dispositivos móviles como *tablets* y *smartphones*, están teniendo mucho éxito los *Tower defense*, estos juegos, a diferencia del resto de videojuegos de estrategia, son bastante más sencillos y no requiere que se tengan en cuenta tantos aspectos y se pueden jugar mucho más rápidamente, sin tener que dedicarles demasiado tiempo, ideales para jugar durante un viaje o en descansos cortos.

En este tipo de juegos el jugador debe encargarse de proteger una estructura mientras hordas de enemigos avanzan con intención de destruirla, el jugador debe poner trampas y unidades en el camino de los enemigos para eliminarlos e intentar que la estructura se mantenga en pie el mayor tiempo posible.

Figura 43. Plants vs Zombies (izquierda) y Final Fantasy Crystal Chronicles: My Life as a Darklord (derecha)

3.6. Velocidad

3.6.1. Simulador

Principalmente son videojuegos en los que se dedican a comenzar de un punto y llegar a una meta antes que los demás participantes. Juegos así se han desarrollado de su forma común, en vehículos, hasta otras formas como juegos de plataformas. La idea principalmente es competir en llegar el primero.

Los simuladores de carreras representan con exactitud las carreras de la actualidad, seguido por variaciones en detalles y agregados. Intentan imitar a la perfección los circuitos de la realidad los sonidos de los vehículos, la potencia, etc.

En este Género destacan juegos como Gran Turismo, *Toca Touring Car, GTR, Forza motorsport, Colin McRae Rally,* y todos aquellos basados en la fórmula uno.

Figura 44. Gran Turismo 6

3.6.2. Arcade

Los videojuegos incluidos en el subgénero de velocidad arcade, cumplen con el mismo objetivo que los de simulación, pero estos no intentan parecerse a la realidad. También consisten en una carrera de punto "A" a punto "B", pero con todo tipo de trampas e irregularidades en el transcurso de la carrera, así como magias, turbos o ayudas para los vehículos.

Las características principales de los videojuegos arcade de velocidad, es que recuerdan a un juego de máquina recreativa, tienen mucha acción en la que se premian los reflejos del jugador e intentan representar la máxima sensación de velocidad y de peligro.

En este género destacan juegos como *Mario Kart, Diddy Kong Racing, OutRun, Wipeout, F-zero.* Son videojuegos que no representan la realidad, pero sí una caricatura de ella.

Figura 45. F-zero para SNES (izquierda) Outrun para Arcade (derecha)

3.7. Sandbox

Los videojuegos del género *Sandbox* no deben ser confundidos con un mundo virtual o un chat general, como *Second life* o *Habbo Hotel.* Los videojuegos *sandbox* son aquellos juegos que no son lineales, ya que no tienen una línea de juego definida por la que el jugador deba avanzar, el orden de las acciones que realiza el jugador es como el jugador desee realizarlas, esto proporciona mayor libertad a la vez que desorientación del jugador, igualmente el juego definirá ciertos objetivos para intentar guiar al jugador.

Una de las características comunes en este género es que se intenta no limitar en nada al jugador y así el jugador es capaz de alterar el entorno de juego. Los juegos más conocidos de este género son juegos como Minecraft, SimCity o GTA.

Figura 46. MineCraft para PC

3.8. Otros

3.8.1. Casual

Con el éxito de dispositivos portátiles como las *tablets* y los *smartphones*, han proliferado un tipo de juegos pensados para este tipo de plataformas, con mecánicas más sencillas y tiempos de juegos más cortos y con controles que intentan aprovechar el tipo de uso propio que tienen estos dispositivos, orientado a jugadores no tan especializados como los que solía haber hace unos años.

Muchas veces estos juegos se basan en alguno de los géneros clásicos de videojuegos, pero simplificando mucho sus normas, de forma que sea posible aprender a jugarlos en un corto periodo de tiempo.

Aunque al principio estos juegos eran exclusivos de dispositivos móviles, su popularidad y el avance en tecnologías captoras de movimiento, aprovechadas en plataformas como Wii, Kinect o PS Move ha hecho que este género salte a otras plataformas y tenga mucho éxito. Actualmente hay juegos de tipo casual que toca casi todo tipo de tema, como la estrategia militar o la música.

Figura 47. Kinect Star Wars (izquierda) y Angry Birds (derecha)

3.8.2. Minijuegos

El subgénero de minijuegos incluye aquellos videojuegos que son de corta complejidad que se encuentran dentro de otros videojuegos o agrupados varios minijuegos en un mismo juego. En ocasiones se utilizan los minijuegos para promocionar un juego o personaje de juego.

Los minijuegos también se conocen como *party games* porque muchos de ellos están pensados para jugar con más gente, sin mucha complejidad para no tener que gastar tiempo en aprender las normas de juego y rápidamente terminar la partida para jugar la revancha.

En este subgénero destacan juegos como los *Mario Party* o *Rayman Rabbits.*

Figura 48. Mario Party para Wii

3.8.3. Puzzle

Los videojuegos de este tipo requieren que el jugador utilice la lógica para resolver acertijos o naveguen por estructuras complejas como laberintos, muy frecuentemente los juegos de puzzles se cruzan con algún otro tipo de género para hacerse más atractivos al público.

Este tipo de juegos han existido desde el principio de la historia de los videojuegos, ya que uno de los ejercicios más interesantes de la informática ha sido el portar problemas de lógica a sistemas informáticos, ya sea para un propósito lúdico o experimental. Además, al principio del desarrollo de los videojuegos, cuando no era requisito que éstos tuvieran una narrativa complicada, era una forma fácil de crear un videojuego sin tener que dedicar mucho tiempo a la planificación del sistema de juego.

Actualmente este género suele estar fuertemente ligado al género casual y podemos ver una gran cantidad de estos juegos en los mercados virtuales de aplicaciones como *Google Play* o *App Store*.

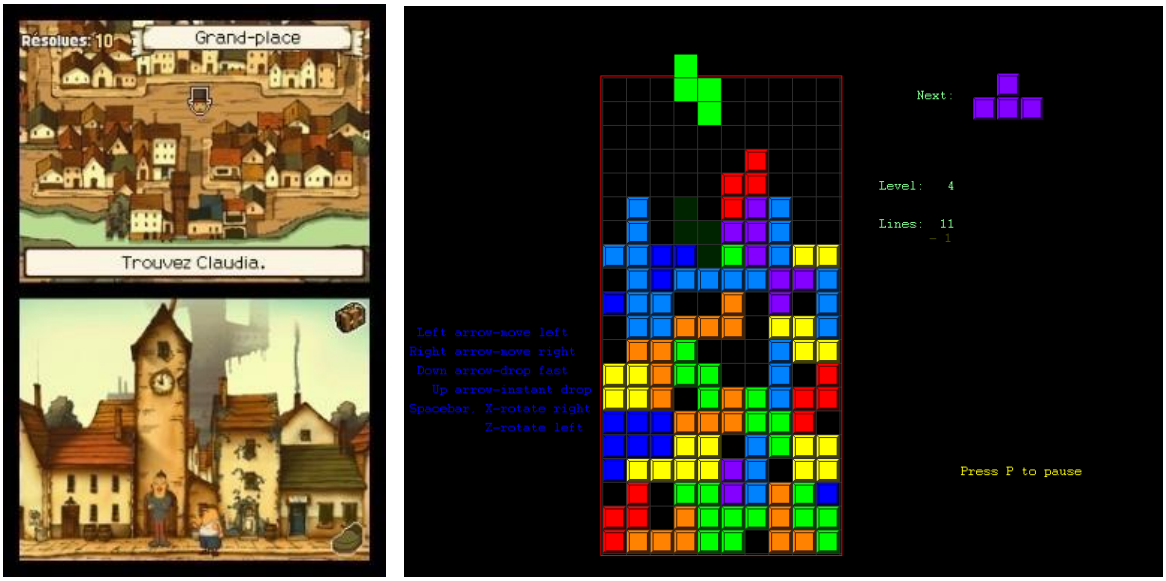


Figura 49. Profesor Layton (izquierda) y Tetris (derecha) juegos que basan su jugabilidad en puzzles

3.8.4. Tablero

Una de las formas más fáciles de desarrollar un videojuego sin tener que pensar su desarrollo es portar un juego de mesa a un dispositivo electrónico, es por eso que desde el inicio del desarrollo de los videojuegos se han podido encontrar juegos que han imitado los mecanismos de juegos de mesa clásicos como el parchís o el ajedrez o juegos populares como Monopoly, aunque estos videojuegos son bastante populares, no suelen ser grandes éxitos, ya que los jugadores suelen buscar experiencias más originales o más orientadas a las plataformas a las que estén jugando.

Pese a todo, estos juegos tienen bastante salida como videojuegos multijugador que complementan a programas de mensajería o redes sociales.

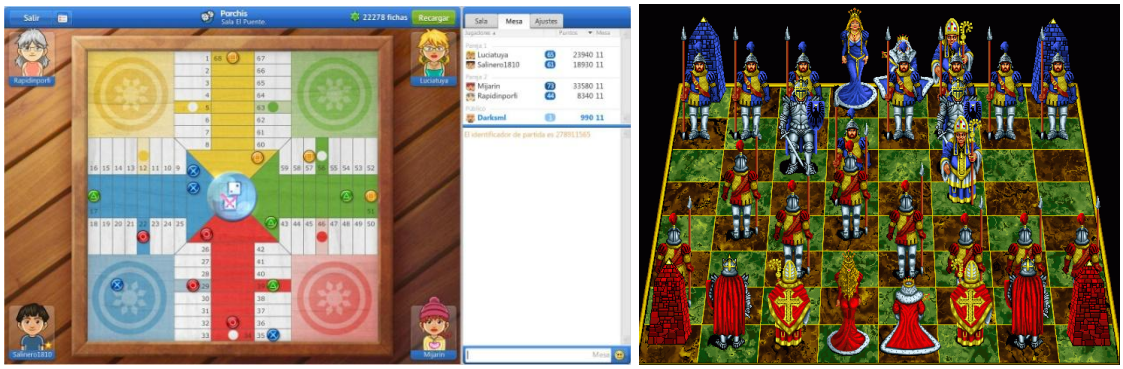


Figura 50. Videojuego de parchís (izquierda) y de ajedrez (derecha)

3.9. Por propósito

3.9.1. Publicitarios

Los videojuegos publicitarios son juegos cuyo propósito es promocionar algún producto o a alguna marca. Se considera que el primer videojuego publicitario fue Chex Quest, que se regalaba con los cereales Chex. En aquel momento no era habitual este uso para los videojuegos, y, aunque algunas multinacionales volvieron a utilizar esta práctica, no fue común hasta la popularización de internet.

Hoy en dia es muy habitual que cualquier marca tenga en su página web oficial algún juego de tipo casual con el propósito de llamar la atención a los visitantes y dar a conocer sus productos, con la proliferación de las redes sociales, se suelen conectar éstas con los juegos promocionales en un intento de hacer viral su promoción.

Figura 51. Chex Quest, primer juego publicitario

3.9.2. Serious games

Los *serious games* son juegos diseñados para otro objetivo distinto al de la pura diversión. Normalmente, el adjetivo "serio" pretende referirse a productos relacionados por industrias como la de defensa, exploración científica, sanitaria, urgencias, planificación cívica, ingeniería, religión y política.

Estos juegos acostumbran a apoyarse en entidades funcionales diferentes a las comunes, como pueden ser brazos mecánicos, radares y vehículos. Usando la infraestructura existente, los desarrolladores de videojuegos pueden simular batallas, procedimientos, eventos o pilotar vehículos complejos por una cantidad económica mucho inferior a la que equivaldría hacerlo en la realidad.

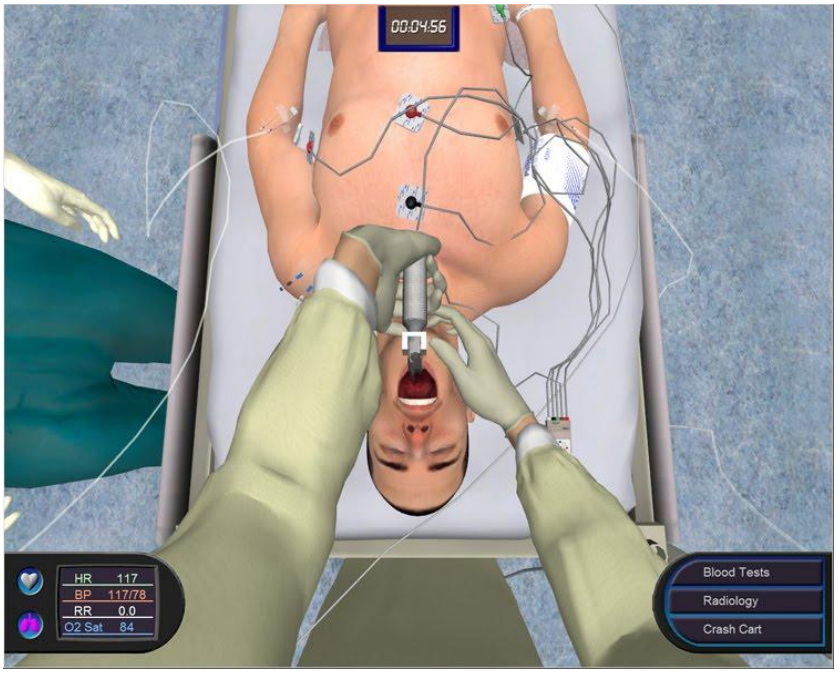


Figura 52. Nevermind

3.9.3. Educativos

La voluntad de diferentes sectores de distanciarse de los videojuegos cuando estos se acercan al ámbito educativo ha propiciado la creación de la categoría de juegos ludo-educativos. Los videojuegos han dejado de estar estigmatizados en el ámbito científico, pero no socialmente. Los juegos que reciben el apelativo de educativos al considerarlos desde la jugabilidad podrían

pertenecer a las categorías antes mencionadas. Solo cuando atendemos a su uso como herramientas educativas tiene sentido establecer una clasificación según:

- La estructura: tutorial, base de datos, simulador, constructor o herramienta.
- Los objetivos educativos que pretende facilitar: conceptuales, procedimentales, actitudinales.
- Las actividades cognitivas que activa y desarrolla: control psicomotriz, observación, memorización, evocación, comprensión, interpretación, comparación, análisis, síntesis, cálculo, razonamiento (deductivo, inductivo, crítico), pensamiento divergente, resolución de problemas, relación (clasificación, ordenación), creación, exploración, experimentación, reflexión metacognitiva, valoración, imaginación…
- Su función en la estrategia didáctica: entrenar, instruir, informar, motivar, explorar, experimentar, expresarse, comunicarse, entretener, evaluar, proveer recursos.

Al igual que todo juego considerado educativo puede estar dentro de las clasificaciones antes mencionadas, todo videojuego en su sentido más amplio también puede ocupar un espacio dentro de esta nueva clasificación, ya que todo juego puede ser asociado a unos objetivos de aprendizaje.

La aplicación de nuevos medios de comunicación en la enseñanza ya inició el debate de qué podemos considerar medios en la enseñanza. Rossi y Biddle (1970), al hablar de los medios de comunicación en la enseñanza, establecieron una conceptualización como "cualquier dispositivo o equipo que se utiliza para transmitir información entre las personas". Sobre esta categorización de carácter general matizan el carácter del "medio educativo" en razón de su utilización para fines educativos, al margen del contexto en el que funcione, refiriéndose a si es en casa o en la escuela.

Para la propagación de este nuevo subgénero o forma de uso de los videojuegos existen diversas páginas web en internet, en las que es posible jugarlos. Varios juegos que se incluyen en este género son juegos como Monturiol, Peratallada, Crusafont, Ferran Alsina o 1714 *the game.* También existen videojuegos no necesariamente orientados a la educación pero dependiendo del uso que se les dé pueden ser muy instructivos como todos los que se basan en historia real para su desarrollo.

Figura 53. Miquel Crusafont (izquierda) Ferran Alsina (derecha)

3.9.4. Juegos de hacer ejercicio

Durante los años 80, en un intento de hacer más atractivo el mundo del videojuego a las mujeres, salieron al mercado algunos juegos cuyo objetivo era el de hacer ejercicio. El experimento no tuvo éxito, en gran parte por las limitaciones tecnológicas de la época. Hasta el día de hoy, han ido saliendo algunos intentos de traer el ejercicio al mundo del ocio electrónico, pero estos intentos han ido pasando por la historia sin pena ni gloria. No ha sido hasta esta última generación de videoconsolas, que con ayuda de las nuevas tecnologías capaces de captar el movimiento del jugador, se han empezado a tener éxito en este campo.

Periféricos como Wii Balance Board o Kinect han permitido hacer videojuegos que de verdad "vean" como el jugador hace el ejercicio y le corrijan si se equivoca, así como hacer un seguimiento de sus progresos. Estas nuevas funcionalidades han hecho que este nuevo intento de hacer ejercicio desde casa haya tenido más éxito que los anteriores, hasta el punto de que se han sacado nuevas versiones mejoradas y se tienen planificados nuevos títulos de este tipo para la siguiente generación.

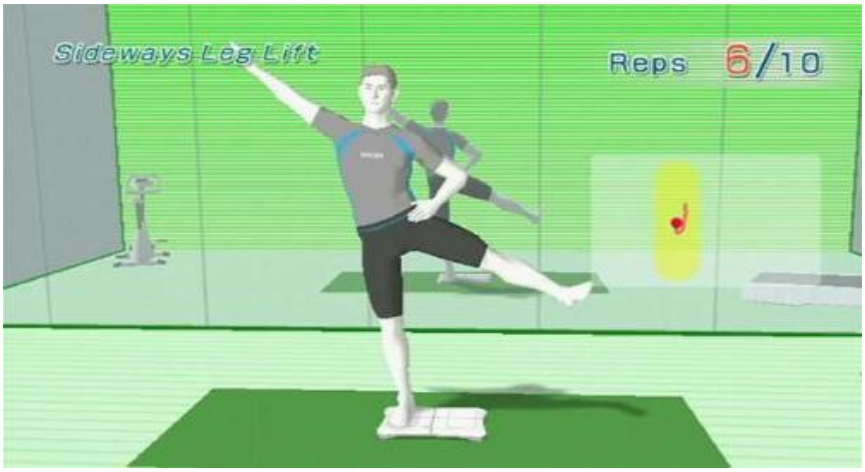


Figura 54. Wii Fit, actualmente el juego más popular para hacer ejercicio

4. Elementos de un videojuego

Un videojuego se compone de muchos elementos, cuando nos hablamos de estos elementos, no nos referimos tan sólo a aspectos funcionales, sino que también hablamos de elementos narrativos o de jugabilidad.

4.1. Personajes

Los personajes son los individuos que pueblan el mundo en el que se sitúa el videojuego y gracias a ellos se desarrolla la historia, según la importancia que tengan estos personajes en el videojuego, tendrán unas funcionalidades u otras.

4.1.1. Protagonistas

El personaje protagonista es el centro de la historia y quien lleva el peso narrativo, alrededor del protagonista o protagonistas suceden todos los incidentes que enlazan la historia sobre la que trata el juego. Cabe destacar que, aunque el protagonista suele ser un personaje jugable, esto no significa que siempre deba ser así, es muy común que haya personajes jugables no protagonistas y también hay algunos ejemplos de videojuegos en los que el protagonista no es personaje jugable.

El protagonista suele tener habilidades fuera de lo común que le ayudarán en su aventura, estas habilidades dependerán del género al que pertenezca el juego en cuestión y las funcionalidades que tenga como personaje jugable dependerá de éstas habilidades, por ejemplo, en un juego de plataformas, el protagonista será capaz de saltar grandes distancias con facilidad, en un juego de lucha, el protagonista será un experto en artes marciales o en una aventura gráfica el protagonista destacará por su ingenio.

4.1.2. Personajes jugables

Personajes jugables son todos los personajes que el jugador manejará, suelen ser los protagonistas, aunque no son infrecuentes los videojuegos en los que se puede controlar a algún personaje secundario, como en los juegos de lucha, en los que el jugador puede elegir a una amplia gama de personajes a los que controlar, no todos protagonistas.

Las funcionalidades necesarias para controlar correctamente a un personaje jugable dependerá según el videojuego, pero en la mayoría de casos será necesario que el personaje se mueva correctamente por el escenario y sea capaz de interactuar con él.

En la siguiente figura vemos en *screenshot* de *The Secret of Monkey Island*, en el que, en la parte inferior, podemos ver las acciones que es capaz de hacer el personaje jugable, en este caso protagonista.

Figura 55. Monkey Island

Como vemos, el protagonista ha de ser capaz de llevar a cabo diferentes acciones con todos los elementos del escenario. En cambio, en la próxima figura vemos otros caso en el que el personaje jugable, en este caso un personaje secundario, tiene otro tipo de funcionalidades, bastante más limitado.

Figura 56. Kingdom Hearts: Birth by Sleep para PSP

Como puede verse, las funcionalidades y características de un personaje jugable son muy variables dependiendo del videojuego en el que nos estemos centrando.

4.1.3. PNJ

Los PNJ recibe sus siglas de la expresión Personajes No Jugadores y, tal como su nombre indica, son personajes que aparecen en un videojuego pero de los cuales el jugador no tiene control sobre ellos, PNJ hay varios tipos y sus funcionalidades dependerán de cual sea su objetivo dentro del videojuego, a continuación vamos a hacer un repaso a algunos de los tipos más comunes de PNJ.

4.1.3.1. Aliados

Estos PNJ suelen ser personajes secundarios del videojuego que ayudan al protagonista o personaje jugable durante algún momento. Los aliados también pueden ser personajes jugables, en cuyo caso no serían PNJ, pero también es muy común que el personaje no tenga control sobre el aliado o tenga un control muy limitado, y tan solo pueda dar unas directrices de cómo se comportará, y dependa de la Inteligencia Artificial del videojuego dar las órdenes que obedecerá el aliado.

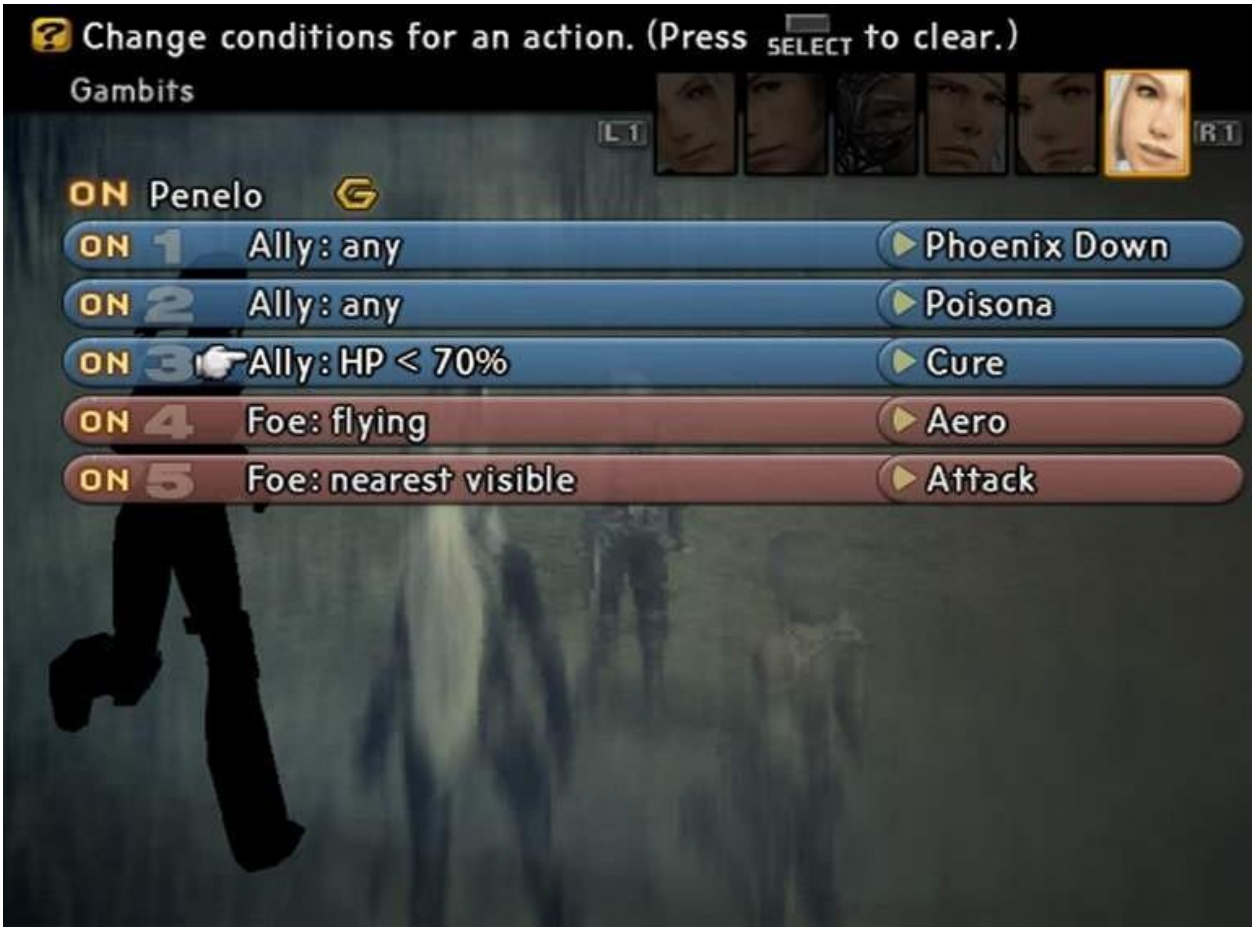


Figura 57. Final Fantasy XII

En la imagen que encabeza estas líneas, podemos ver un *screenshot* de *Final Fantasy XII*, un videojuego en el que los aliados son personajes no jugables controlados por la IA, pero en el que el jugador tiene capacidad de crear unas normas que obedecerá el aliado. En esta *screenshot* en particular vemos las reglas que se han definido para un personaje en concreto.

4.1.3.2. Relleno

Estos PNJ no tienen más funcionalidad que la de estar en el escenario para crear la ilusión de que un escenario en concreto está muy discurrido, estos personajes no suelen tener ninguna otra funcionalidad, y si se intenta interaccionar con ellos tan solo tendrán una o dos frases en las que hacen algún comentario para dar más vida al lugar que se esté representando en ese momento.

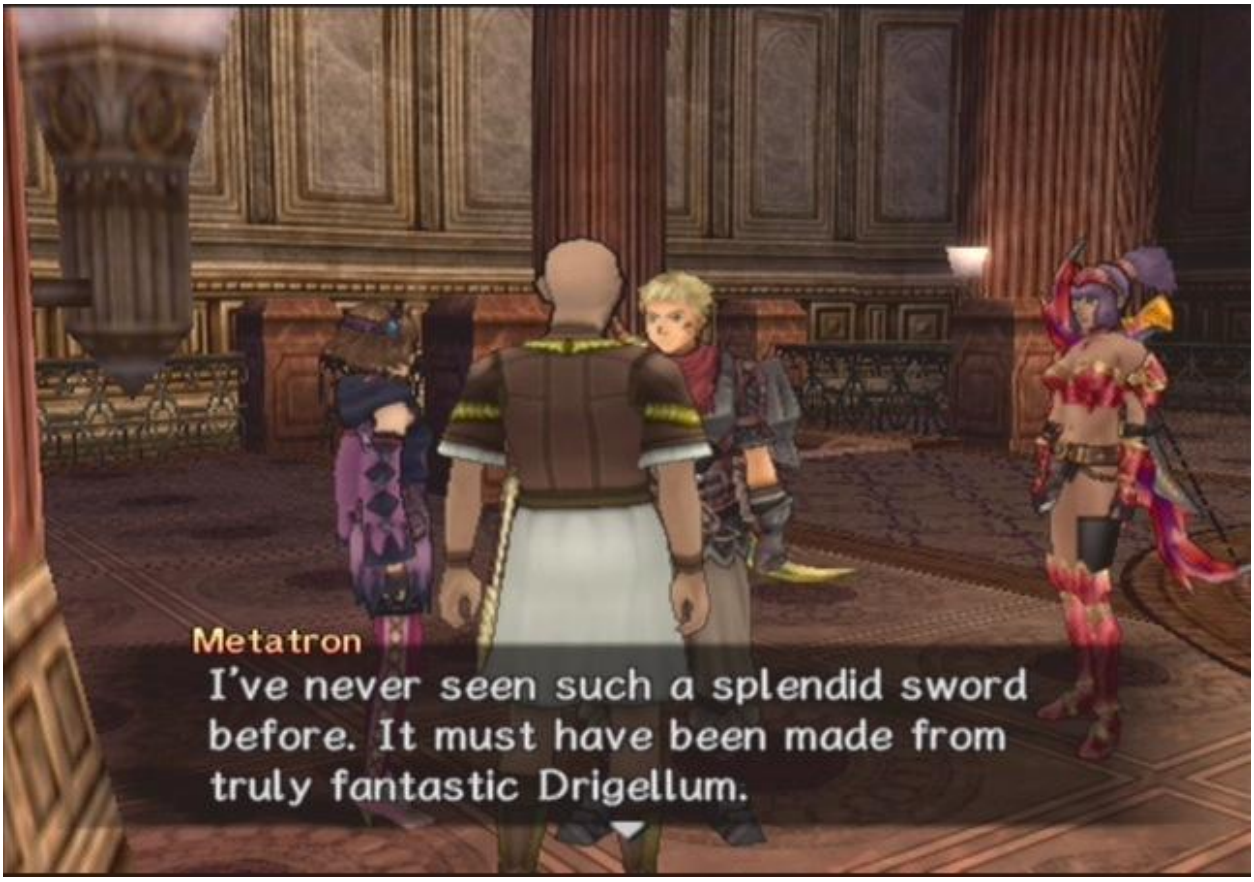


Figura 58. El protagonista de Rogue Galaxy hablando con un PNJ de relleno

4.1.3.3. Quest givers

Estos personajes son los que se encargan de dar tareas a los jugadores que deben cumplir, todos los videojuegos tienen algún elemento que se encargue de dar objetivos, pero los más comunes son los PNJ. Los Quest givers pueden ser personajes importantes para el desarrollo de la historia o pueden ser personajes creados específicamente para dar esa tarea al jugador y después perder toda importancia dentro del juego, convirtiéndose entonces en un personaje de relleno.

Muchas veces estos personajes son también los que entregan la recompensa adecuada al jugador una vez que ha completado la tarea.

Los quest givers son especialmente distinguibles en los MMORPG, ya que estos suelen centrarse precisamente en la resolución de tareas cerradas y está claramente indicado qué personajes te darán tareas adicionales.

Figura 59. Un jugador recibiendo una tarea de un PNJ

4.1.4. Enemigos

4.1.4.1. Comunes

Los enemigos comunes son aquellos que continuamente atacarán a nuestro protagonista, aunque para ello tengan que morir en el intento. Su principal misión es la de evitar que el protagonista llegue al jefe principal del juego. Los hay de muchos tamaños y tipos, y algunos otorgan algún que otro premio.

Las características principales de estos enemigos, es su sencillez a la hora de eliminarlos y su sencilla inteligencia artificial. Acostumbran a moverse en grupos y suelen aparecer numerosas veces en varios escenarios. Son controlados por la CPU y en muchas ocasiones permiten al protagonista incluso ignorarlos. Suelen aparecer especialmente en el género de plataformas y en los *beat'em ups*.

Este tipo de enemigos no acostumbran a tener mucha personalidad porque en su vida solo tienen un objetivo obedecer órdenes, eliminar al protagonista o defender a su jefe. Es posible que algún enemigo común previamente haya sido un minijefe que posteriormente encontremos como enemigo común defendiendo a un jefe. Hay muchos juegos que utilizan enemigos comunes como

método de entretenimiento o de *crawleo* para conseguir objetos como la saga Mario, la saga Sonic, la saga Kirby, devil my cry, God of War.

Figura 60. Koopa Troopa de la saga Mario (izquierda), enemigo Kirby (derecha)

4.1.4.2. Jefes

En los videojuegos un jefe o monstruo, es un personaje particularmente desafiante, controlado por la CPU y dotados de una inteligencia artificial. Este enemigo debe ser vencido en algún punto del juego. Los jefes finales aparecen en la mayoría de géneros de juego, especialmente en el género de plataformas y en los *beat'em ups*.

Este tipo de enemigos adoptan el concepto de los antagonistas de las historias, que lideran grupos de enemigos, monstruos o malhechores, estos enemigos comunes pueden haber sido anteriormente un minijefe. Un jefe puede no tener relación en el protagonista en si o con el resto de enemigos. Un jefe se caracteriza por ser más fuerte y difícil de vencer que el resto de los enemigos del videojuego. Usualmente los jefes van acompañados de súbditos que su único objetivo en la vida es defender a su jefe de los ataques del protagonista.

Normalmente cuando el protagonista vence a un jefe, puede recibir un premio o una nueva arma, una reliquia o simplemente culmina el nivel. La resistencia física del jefe es mayor a la de los enemigos comunes, por ello se representa con una barra de vida o número de puntos de resistencia que disminuye conforme el jefe vaya siendo atacado por el jugador, normalmente este jefe también va mejorando su combate conforme va perdiendo mas vida.

Al enfrentarse contra un jefe, generalmente cambia la ambientación: la música se torna tétrica o de suspense, la iluminación es más dramática y el escenario en sí es diferente al del nivel habitual, suelen ser nuevos circuitos cerrados en los que no se puede escapar del jefe. En ocasiones la única forma de vencer al jefe es utilizando combos, combos especiales o algún arma que se activa al comenzar la batalla. Para matar a un jefe se acostumbra a necesitar una estrategia o adquirir poderes especiales solo para esta batalla.

Figura 61. Sefirot de Final fantasy VII (izquierda) Eggman de la saga Sonic (derecha)

4.2. HUD

Se llama HUD (siglas de Heads-Up Display) a la información que aparece durante toda la partida en algunos juegos, ya sea en forma de números o de icono y que indica al jugador todo lo que deba saber para poder. continuar con la partida.

La complejidad de un HUD puede variar según el juego que se trate, puede ser algo muy simple como un número indicando la puntuación o las vidas restantes, como es el caso de Super Mario Bros o tan complejos como el caso de Halo, en el que se indican los puntos de vida restantes, totales, el arma equipada, la munición y mucha más información.

Un HUD debe ser diseñado para no entorpecer el juego y mostrar en todo momento información útil al jugador, por lo general, se ha comprobado que la posición más cómoda para representar la información es en los bordes y las esquinas de la pantalla, de forma que el jugador pueda consultar esa información con solo mover el ojo.

Figura 62. HUD de Halo

Hay videojuegos que, con tal de mantener el ambiente de juego, presentan el HUD como si fuera un verdadero HUD del mundo del juego, como muchos juegos de conducción en primera persona, por ejemplo, otros en cambio, optan por mostrar el mínimo de información necesaria para dar sensación de incertidumbre sobre el estado del personaje jugable, dando tan solo pistas, como por ejemplo enrojeciendo los bordes de la pantalla.

4.3. I.A

La inteligencia artificial en un videojuego, se refiere a las técnicas utilizadas para producir la ilusión de cierta inteligencia en el comportamiento de ciertos personajes no jugables PNJ. El propósito de la inteligencia artificial es evaluar la situación actual de la partida y actuar como si fuera un jugador sin ventajas de datos a los que no pueda acceder el jugador.

Como la IA se centra en el aspecto de la inteligencia y buena jugabilidad, su enfoque es muy diferente a la de la IA tradicional, soluciones alternativas y trucos son aceptables, sus habilidades pueden bajar el tono para dar a los jugadores un sentido de justicia. Esto, por ejemplo, se usa en

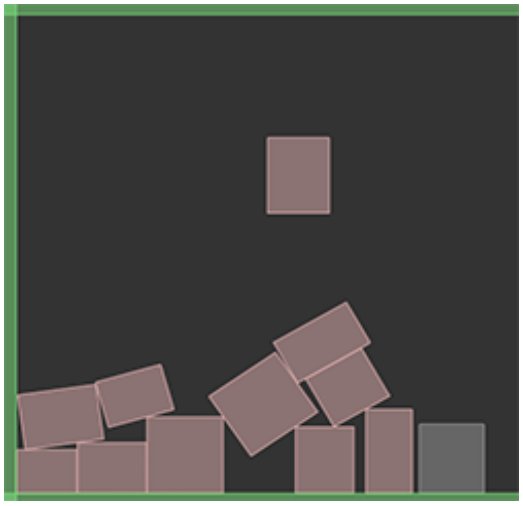
los videojuegos del género *shooter*, donde la puntería perfecta estaría fuera de lo que un humano pueda alcanzar.

La inteligencia artificial se utiliza en varios campos de los videojuegos aunque la mas evidente es en el control de los PNJs. También se puede utilizar para buscar rutas los *pathfinders*, más utilizada en los videojuegos de estrategia para llegar de un punto "A" a un punto "B" evitando obstáculos. También la navegación de los PNJs de navegar a su entorno como si fuera transeúntes. La IA también se encarga de equilibrar la dificultad del juego para adaptarla a cada jugador y que cada jugador pueda exprimir al máximo sus cualidades.

4.4. Física

La física en los videojuegos es un tipo de programación, en la que se introducen las leyes de la física en un motor de juego. El propósito es hacer que los efectos físicos de los objetos modelados tengan las mismas características que en la vida real, teniendo en cuenta, por ejemplo, gravedad, masa, fricción, restitución, etc.

La simulación de la física en la programación es solo una aproximación a la física real, y el cómputo es desarrollado usando valores discretos, Hay muchos elementos que forman los componentes de la simulación física. Por ejemplo en el caso de nuestro motor para el cálculo de la gravedad nosotros utilizamos una relación directa entre una sinusoidal y el tiempo que está el personaje en el aire.



Hay muchos tipos de física, física grabada que consiste en grabar una animación y reproducir siempre el mismo comportamiento independientemente del entorno, o calcular a cada momento, la posición del objeto respecto a todos los valores comentados anteriormente (gravedad, masa, fricción, restitución, etc). También existe la posibilidad de contemplar todos esos cálculos pero en cada hueso en los que está compuesto el personaje y a esto se le llama *Ragdoll*.

Figura 63. Física aplicada a un conjunto de cajas

4.5. Online

Son aquellos videojuegos que se juegan vía internet. Acostumbran a ser videojuegos multijugador en los que te encuentras con otro rival que puede estar en cualquier parte del mundo.

En la actualidad la mayoría de los videojuegos disponen de esta opción, también se están adaptando a las redes sociales con opciones como subir trofeos, logros o incluso videos de como has logrado completar un escenario.

También se están popularizando los videojuegos online porque se realizan muchas competiciones para diferentes videojuegos, en las que se reúnen varios equipos desde diferentes partes del mundo en una partida para lograr la victoria de la competición y así conseguir algún tipo de trofeo.

La gran mayoría de los videojuegos modernos disponen de esta opción de acceder a la red para buscar rivales pero no cabe destacar que los más famosos sin duda son juegos como Counter Strike, League of legends, Starcraft e incluso juegos para dispositivos móviles como triviados.

Figura 64. Counter Strike 1.5 para PC(izquierda) y League of Legends para PC (derecha)

4.6. Narrativo

Hoy en día la gran mayoría de los videojuegos tienen un fuerte componente narrativo, aunque no es imprescindible y aún hoy podemos encontrar juegos muy populares que no tienen ningún tipo de historia detrás y van directos a la jugabilidad, pero esto no es lo habitual, normalmente tienen algún tipo de historia, aunque esta sea muy simple y solo sirva de excusa para dar paso al juego, como el caso de Angry Birds.

Suelen utilizarse distintos recursos para representar la historia que está viviendo el personaje jugable, estos recursos son los siguientes:

4.6.1. Textos explicativos

La primera forma que se utilizó para narrar la historia de un videojuego fueron textos explicativos, que al principio de la partida ponían al jugador al corriente de los acontecimientos y motivaciones del protagonista.

En los primeros videojuegos normalmente sólo había una al principio y luego no volvía a haber más referencias a la historia durante el resto del videojuego, hoy en día lo habitual es que estos textos surjan varias veces durante la partida para explicar los controles y las mecánicas de juego al jugador, en lugar de explicar los sucesos de la historia.

4.6.2. Diálogos

Una forma muy habitual de dar salida al contenido narrativo de un videojuego son los diálogos. Conversaciones entre dos personajes que ayudan al jugador a situarse dentro de la historia y el mundo del videojuego.

Al principio la forma de mostrar los diálogos era con cuadros de texto, muy parecido a los textos explicativos, pero a diferencia de los textos explicativos, los diálogos permiten que el jugador se sienta más identificado con el personaje jugable, ya que no solo se pone en su situación durante el juego, sino que tiene la misma información que él y los futuros acontecimientos tienen el potencial de sorprender al jugador igual que al personaje jugable.

Con la mejora de la potencia de las plataformas de juego, se pudieron incluir grabaciones de voz en los videojuegos, de forma que, de forma parecida al cine, ya no era necesario leer las conversaciones, sino que se podían oír. Actualmente es habitual que en los videojuegos de mayor presupuesto se cuente con las voces de celebridades para interpretar a los protagonistas.

4.6.3. Cinemáticas

Las cinemáticas son las secuencias de video durante las cuales el jugador tiene un control nulo o limitado y que se utilizan en los videojuegos para avanzar en la trama, presentar personajes o dar ambientación.

Las cinemáticas se pueden clasificar en tipos, según la forma en la que se generen, existen las cinemáticas con imagen real, en las que se utilizan actores reales para grabar escenas. Estas escenas se reproducen luego en el videojuego para narrar la historia. Este tipo de escenas fueron

muy populares durante la década de los 90, debido a la popularización del CD-ROM y el espacio extra que proporcionaba en contraste a los discos flexibles que se utilizaban hasta aquel momento. Aún así, estas escenas no se perpetuaron, debido a que rompían la estética del juego.

Figura 65. Phantasmagoria, aventura gráfica de Sierra Online que utilizaba escenas con imagen real

La alternativa a las escenas con imagen real son las escenas animadas, de estas hay dos tipos, la escena pre-renderizada y las renderizadas en tiempo real. Estas últimas fueron las primeras en utilizarse, ya que para prepararlas tan solo había que programar un *script* con las acciones que debían realizar los personajes y los textos que se tenían que mostrar, utilizando el mismo game engine que ya utilizaba el videojuego.

La calidad de estas escenas era escasa, por lo que proliferó el uso de las escenas pre-renderizadas, estas escenas estaban animadas y renderizadas por los desarrolladores del juego y tenían la ventaja de que podían mostrar escenas de mayor calidad y los personajes podían realizar acciones que el motor no permitía, pero sin romper tanto la estética como lo hacían las escenas con actores reales.

El inconveniente que tienen este tipo de cinemática es que si la diferencia de calidad gráfica entre el *game engine* del juego y la escena pre-renderizada era muy grande podía haber dificultades para reconocer a los personajes. Con el aumento de la complejidad de los videojuegos se introdujo la posibilidad de cambiar el aspecto de los personajes, lo cual trajo el inconveniente de que los protagonistas cambiaban de aspecto durante estas escenas. Este inconveniente y el aumento de la potencia gráfica de las plataformas de juego ha hecho que en los últimos años muchas desarrolladoras hayan optado por volver a las escenas renderizadas en tiempo real.

El regreso a las escenas renderizadas en tiempo real han propiciado que las cinemáticas sean interactivas y que permitan una limitada interacción por parte del jugador, normalmente las acciones que se pueden hacer durante estas escenas son mínimas y no tienen impacto en el desarrollo del juego, aunque en los últimos tiempos algunas desarrolladoras se han centrado en crear juegos que se basan totalmente en cinemáticas, como *Quantic Dream*, que da más liberado al jugador y cuyas acciones hacen que el desarrollo del juego cambie radicalmente. Por supuesto, esto hace muy complicado el desarrollo del juego, ya que se han de crear cinemáticas para todas las posibles acciones del jugador.

Figura 66. Beyond: Two Souls, último juego desarrollado por Quantic Dream

4.7. Hardware

4.7.1. Periféricos

Los periféricos son controladores de entrada utilizados para enviar señales a un videojuego. Un controlador está conectado normalmente a una videoconsola o a un PC. Hay muchos tipos de periféricos para conectar, diseñados para jugar. Hay algunos dispositivos genéricos como el teclado y el ratón o el Gamepad y otros más específicos como un volante o un apistola.

Diferentes controladores:

- Los **Arcade Sticks**: Tienen varios botones y uno o más joysticks, que permiten al usuario mover el personaje por la pantalla y ejecutar acciones mediante el uso de los botones. Pueden ser consideradas como una combinación entre el gamepad y el joystick, pero son las precursoras de ambos.

Figura 67. Arcade Stick diseñado para Tekken 6

- Los **Gamepads**: El gamepad, también conocido como joypad, es un tipo de controlador de videojuego que se sujeta con las dos manos, de manera que los pulgares se usan para la entrada de datos. Los gamepads suelen tener una serie de botones de acción (manejados con el pulgar derecho) y una serie de botones de dirección (manejados con el pulgar izquierdo), lo cual es incómodo para las personas zurdas. Muchos de los controladores de juegos modernos son variaciones del gamepad estándar. Algunos Gamepads proporcionan un motor interno que proporciona tecnología de vibración. Los gamepads son el principal método de entrada de las videoconsolas actuales.

Figura 68. Gamepad diseñado para la nueva Playstation4

- **Paddle**: Es un controlador que contiene una rueda giratoria y varios botones de disparo. Es usado normalmente para controlar los movimientos del jugador o de un objeto a lo largo de un único eje de la pantalla. El controlador de paddle fue uno de los primeros controladores analógicos. Han ido desapareciendo conforme los juegos "paddle and ball" han ido perdiendo popularidad.

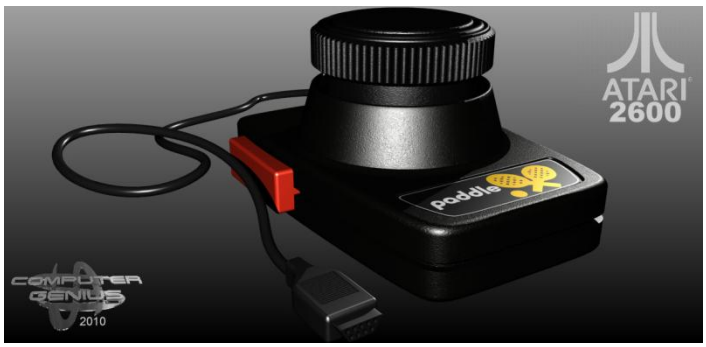


Figura 69. Paddle diseñado por atari

- **Joystick** de **vuelo**: Es un periférico que es similar al control de mando de una aeronave. Consta de una palanca que gira sobre uno de los extremos, transmitiendo el ángulo de giro en dos o tres dimensiones al ordenador. A menudo es usado en simuladores de vuelo. Los controladores HOTAS (hands on throttle-and-stick), que incluyen hardware adicional para simular controles de la válvula reguladora y de timón, son populares entre los fanáticos del género.

Figura 70. Joystick de vuelo para simulador de vuelo

- **Volante**: El volante, esencialmente una versión mayor del paddle, es usado para videojuegos de carreras como Need for Speed, Gran Turismo, Forza Motorsport y Collin McRae Rally. Muchos incluyen la tecnología force feedback, es decir, diseñados para dar la misma sensación que se tienen al conducir un coche real, (dureza en la dirección, tirada en curva, baches etc...) aunque el nivel de realismo alcanzado depende del juego y la configuración del mismo. Normalmente vienen con 2 pedales para controlar el acelerador y el freno, aunque en los volantes más avanzados se incluye un tercer pedal para controlar el embrague. Se puede cambiar de marcha con un paddle o Léva, con una palanca de cambios en modo secuencial que se mueve adelante o atrás o con una palanca en "H" que simula el cambio de .los vehículos reales utilizando un embrague. La mayoría de los volantes giran sólo 200 o 270 grados, pero algunos, como el Logitech Driving Force Pro, Logitech G25, y Logitech G27, pueden girar 900 grados como un vehículo real.

Figura 71. Volante G27 junto con el asiento playseat

- **Teclado** y **ratón**: El teclado y el ratón son dispositivos de entrada típicos de un PC y son actualmente los principales controladores de juegos para ordenadores. Algunas consolas de videojuegos también pueden funcionar con teclado y ratón. El teclado del ordenador está basado en el de la máquina de escribir y fue diseñado para introducir texto escrito. Para juegos, normalmente el teclado se usa para controlar los movimientos del personaje mientras que el ratón es usado para controlar la cámara o para apuntar. El teclado numérico (conjunto de teclas con, al menos, los números del 0 al 9, situados en forma de cuadrado), que forma parte del teclado, es también usado como controlador de juegos. Puede encontrarse en un gran número de dispositivos, como en consolas actuales, y normalmente incrustado en un joystick o en un paddle.

Figura 72. Ratón (izquierda) y teclado (derecha) especializados para HardCore Gamers

- **Pistolas**: Una pistola láser es un periférico usado para "disparar" a objetivos en la pantalla. Suelen recordar armas de fuego reales. Su uso está limitado juegos de tipo "rail shooter". La primera pistola láser fue lanzada por Magnavox Odyssey.

Figura 73. Pistola G-Con diseñada por NAMCO para Time Crisis

- **Instrumentos**: Un instrumento músical básicamente es un periférico con una forma que se asemeja a la de un instrumento real, existen varias adaptaciones como baterías, guitarras eléctricas o incluso los micrófonos.

Figura 74. Grupo de instrumentos musicales para jugar a Rockband

- **Pantalla táctil**: Una pantalla táctil es un dispositivo de entrada que permite al usuario interactuar con el ordenador tocando la pantalla. Nintendo popularizó su uso en los videojuegos con la videoconsola Nintendo DS, otros sistemas como smartphones o tablets incluyen este sistema. Las pantallas táctiles modernas utilizan una fina, duradera y transparente capa de plástico encima de la pantalla de cristal. La localización de la zona que tocamos es calculada por la densidad de carga de los ejes X e Y, la cual varía según en la zona en la que toquemos.

4.8. Jugabilidad

La jugabilidad es un término que sirve para calificar la calidad de un juego en términos de sus reglas y diseño como juego, normalmente ha sido un término utilizado tan solo en videojuegos, pero se ha ido extendiendo a otras áreas como los juegos de mesa.

La jugabilidad es una forma de juzgar cuál será la experiencia de juego del jugador, para ello se tienen en cuenta una serie de propiedades como pueden ser:

- Satisfacción: Es el agrado del jugador ante el videojuego completo o mecánicas concretas de éste.
- Aprendizaje: La facilidad para aprender y comprender el sistema de juego
- Efectividad: Capacidad para entretener y divertir al jugador.
- Inmersión: La capacidad que tiene el videojuego para integrarse en el mundo que se describe y "creerse" la historia que se narra
- Motivación: Este atributo tiene en cuenta la capacidad que tiene el videojuego para dar al jugador el ánimo para completar las tareas que le encomienda.
- Emoción: Este atributo tiene el objetivo de estimular al jugador y provocar reacciones emotivas en éste.
- Socialización: Punto que algunos videojuegos intentan cumplir. Se trata de la capacidad del juego para hacer que los jugadores se relacionen entre sí.

En los últimos años, se han desarrollado algunas mecánicas que añaden complejidad a la jugabilidad y que debido a su éxito se han extendido y actualmente la gran mayoría de videojuegos los incluye de una forma u otra.

4.8.1. Progresión de personaje

Aspecto antiguamente exclusivo del género RPG, hoy en día casi todos los videojuegos lo incluyen. Esta mecánica permite al jugador "evolucionar" al personaje jugable, decidiendo sus fortalezas y creando de esta manera personajes muy personalizables y con estrategias propias.

Cuando se habla de progresión de personaje, se tiene en cuenta las habilidades y competencias que el personaje podrá adquirir durante el transcurso del juego y que pueden cambiar radicalmente la forma de jugar.

Hay videojuegos que una gran personalización que otros, los MMORPG son ejemplos de juegos con gran personalización de personaje, que dan la posibilidad hasta de escoger el aspecto físico del personaje. Otros en cambio, tienen la progresión del personaje muy restringida, como es el caso de los Sandbox, en los que el personaje es estático y el progreso está totalmente pautado durante todo el desarrollo del juego.

4.8.2. Logros

En cierta forma, una expansión de la mecánica anterior, los logros son una especie de metajuego, una forma de premiar al jugador, pero en lugar de darle premios al personaje que controla, se le da al propio jugador. Las últimas plataformas de juego tienen esta funcionalidad, cada juego que sale para esa plataforma tiene una serie de logros, con los que se premia al jugador cuando éste lleva a cabo ciertas tareas, estos logros además pueden venir acompañados de unos "puntos de experiencia" que pueden hacer subir el "nivel del jugador". Todo este sistema recompensa al jugador por jugar y además le da una categoría, lo que le anima a seguir jugando.

A esta mecánica se la ha llamado "*gamification*", y se ha llevado con éxito a otros contextos diferentes a los videojuegos. Actualmente se pueden encontrar sistemas de este tipo en páginas web comerciales, que premian comportamientos como el uso de sus funcionalidades sociales.

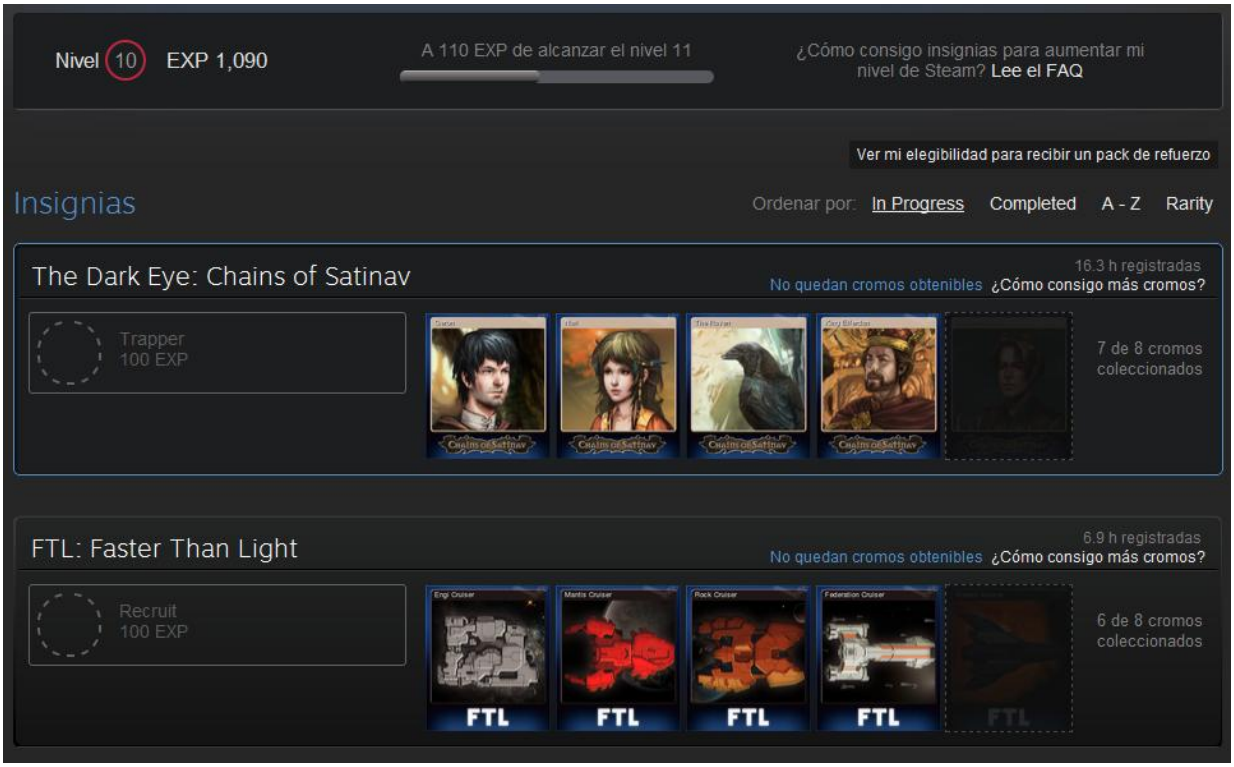


Figura 75. Sistema de Gamification de Steam

5. Tecnologías utilizadas

5.1. HTML5

Figura 76. Logo HTML5

HTML5 es la quinta y última revisión hasta el momento del lenguaje de etiquetas de hipertexto HTML, el estándar utilizado para estructurar y presentar información en la World Wide Web a cargo de W3C.

HTML5 es todavía experimental y no todas sus funciones son soportadas completamente por todos los navegadores y dispositivos, aún así, es utilizado ampliamente por desarrolladores debido a sus avances, que no son difíciles de reproducir de otros modos.

HTML5 incluye nuevos elementos y atributos que reflejan el uso que se hace habitualmente de los sitios webs modernos, y facilitan la implementación de funcionalidades que se han hecho comunes gracias a los sitios web 2.0

En este proyecto HTML5 ha sido indispensable debido a su nuevo elemento <canvas>, que ha sido central en el proyecto y del que se hablará más adelante.

5.2. JavaScript

Javascript es un lenguaje de programación interpretado que se utiliza para implementar sitios web del lado del cliente, permitiendo una serie de mejoras o personalizaciones en la interfaz de usuario y en páginas web dinámicas.

Javascript es un lenguaje orientado a objetos, imperativa, basado en prototipos, dinámico y débilmente tipado. Estas características han hecho que sea fácil trabajar con este lenguaje, ya que es muy parecido a tipos los de lenguajes más habituales, como C, C++ o Java y no ha sido necesario un tiempo de aprendizaje excesivo para poder utilizar la potencia de este lenguaje con soltura.

El uso de javascript en este proyecto ha sido fundamental. Tanto el editor como el *game engine* se ejecutan en el lado cliente, por lo que el javascript ha sido el lenguaje principal con el que se han desarrollado las dos partes del proyecto.

5.3. CSS3

Las hojas de estilo en cascada, o CSS por sus siglas en inglés (Cascade Style Sheet) son un lenguaje que se utiliza para describir la presentación de un documento escrito. Su uso más común es dar estilo a páginas web escritas en el lenguaje de etiquetas HTML o XHTML.

CSS3 es la última revisión que W3C ha hecho sobre este estándar y ha añadido nuevas funcionalidades sobre la anterior, debido a la gran modularización de CSS3, sus diferentes componentes están en distinto nivel de desarrollo.

El uso de CSS3 no es esencial, ya que su uso es exclusivamente estético, pero aún así juega un papel importante, ya que ayuda con la organización del editor y una buena presentación puede disminuir la curva de aprendizaje y facilitar su uso.

5.4. XML

El lenguaje de etiquetas XML, siglas en inglés de eXtensible Markup Language (lenguaje de marcas extensible), es un lenguaje de etiquetas desarrollado por *world wide web consortium* (W3C) utilizado para almacenar datos en forma legible, tanto por humanos como por máquinas.

Este lenguaje deriva de SGML y permite definir la gramática de lenguajes específicos igual que HTML, para estructurar grandes cantidades de datos.

A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil para comunicar varias aplicación que se deben comunicar entre si y utilizan diferentes lenguajes. Esto se adapta perfectamente a nuestro proyecto para permitir comunicaciones entre servidor y editor.

XML no es únicamente para su utilización en internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier tipo de datos.

5.5. JSON

Las siglas JSON son un acrónimo de *Javascript Object Notation,* es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notificación literal de objetos de JavaScript que no requiere el uso de XML para comunicar varias aplicaciones que trabajan con diferentes lenguajes.

JSON es una alternativa a XML gracias a su simplicidad, a la hora de generar información sin etiquetas. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico de JSON, sobre todo si estamos trabajando con JavaScript. Un texto en JSON se puede leer directamente como si este fuera una variable, lo cual ha sido fundamental en este proyecto para escribir los escenarios y leerlos por JavaScript.

5.6. Canvas

Canvas significa lienzo en inglés y es un elemento HTML incorporado a partir de la última versión de HTML, HTML5 que permite la generación de gráficos dinámicamente por medio de un script en JavaSCript. Permite generar gráficos estáticos y animaciones.

Fue implementado por Apple para su navegador Safari. Más tarde fue adoptado por otros navegadores, como Firefox, Opera y Chrome.

El objeto canvas puede ser accedido a través de JavaScript, permitiendo generar gráficos 2D, juegos, animaciones y composición de imágenes. Existe otra etiqueta SVG que cumple con funciones similares, pero no se adapta a nuestro motor, ya que trabaja con vectores y permite hacer imágenes con mayor calidad pero que consumen mucha más potencia del navegador.

5.7. Flask

Flask es un *framework* ligero escrito en *python* y basado en WSGI. Flask es llamado *microframework* porque tiene una base muy simple pero fácilmente extensible. No tiene abstracción de datos en diferentes capas. Existen diferentes librerías ajenas a flask que permiten la abstracción de datos, pero no vienen por defecto, en nuestro caso utilizamos librerías externas para leer XML.

Flask ha sido muy útil en este proyecto para poder programar un servidor en la nube con pocas instrucciones.

5.8. Python

Python es un lenguaje de programación interpretado, su forma de programación obliga al programador a escribir una sintaxis muy limpia y que favorezca un código legible.

Python soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Python es administrado por la *Python Software Foundation.* Un objetivo importante del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en Co C++. Python puede incluirse en aplicaciones que necesiten una interfaz programable.

Este lenguaje ha sido muy útil a la hora de programar el servidor, porque es sencillo y rápido, ayuda al programador a conseguir sus objetivos lo antes posible.

5.9. AJAX

AJAX es una técnica de desarrollo web para la creación de aplicaciones interactivas. Las aplicaciones desarrolladas con AJAX se ejecutan en el lado del cliente, mientras mantienen en segundo plano una conexión asícrona con el servidor. Esta técnica nos permite la modificación de una página web sin necesidad de tener que recargar todo el contenido.

AJAX ha sido muy útil en el desarrollo de éste proyecto, por parte del editor, ha permitido que éste tenga contacto con los recursos utilizados para construir un nivel, así como permitir al usuario subir contenido propio sin necesidad de estar recargando la página continuamente. Por parte del *game engine*, ha permitido comunicar editor y *engine* y de esta forma enviar los niveles construidos por el usuario al *engine* para que éste pueda ejecutarlo.

5.10. jQuery

jQuery es una biblioteca de Javascript publicada como software libre y abierto, licenciado bajo la Licencia MIT y la Licencia Pública General de GNU 2 que permite simplificar la forma de manipular distintos aspectos del desarrollo web en Javascript, como manipular el árbol DOM, manejar eventos o facilitar la interacción con la técnica AJAX.

Javascript es ampliamente utilizado por muchos desarrolladores web e incluso empresas como Microsoft y Nokia han anunciado que incluirán esta biblioteca en sus plataformas y es compatible con la mayoría de navegadores y dispositivos.

El uso de jQuery ha ayudado mucho en el desarrollo del proyecto, ya que ha facilitado muchas funciones como AJAX o el manejo de eventos, que de otra manera habría requerido mucho más tiempo para ser desarrollados.

6. Estructura del proyecto

Este proyecto se ha dividido en dos partes claramente diferenciadas, por una parte el editor, que permite al usuario crear el juego, desarrollada por Rodolfo Núñez del Río y por otra parte el *Game Engine*, el conjunto de rutinas que permiten ejecute el juego una vez terminada la creación de éste, desarrollada por Víctor Manuel Agüero Requena.

En el siguiente esquema podemos ver de forma sencilla como se han organizado los componentes del proyecto.

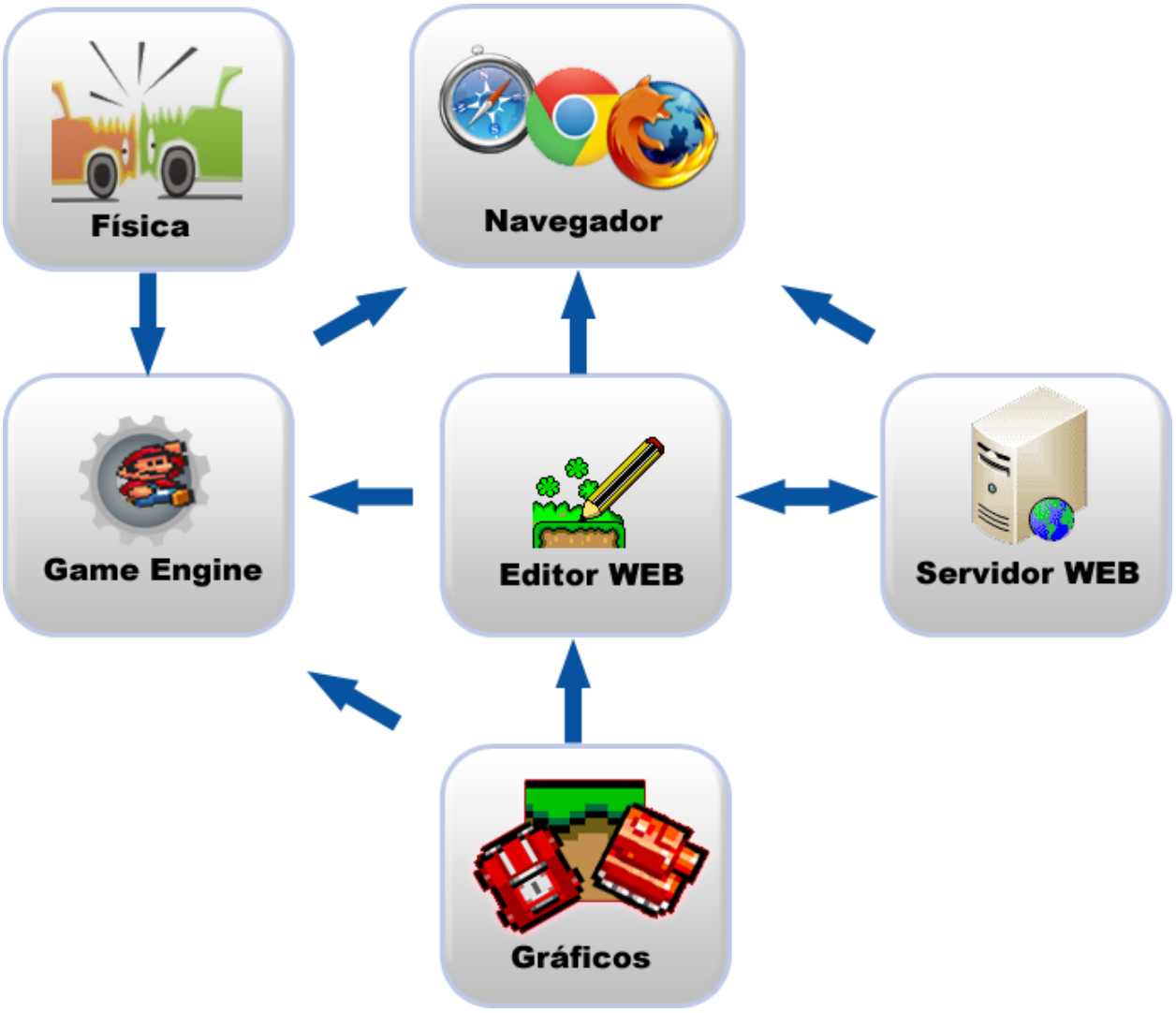


Figura 77. Esquema de la estructura del proyecto

6.1. Game Engine

El *Game Engine* es un conjunto de scripts que controlan todo el funcionamiento interno del videojuego. Desde el movimiento de los personajes hasta las físicas que definen el mundo del juego, pasando por la interacción con el jugador.

Para más información consultar la memoria del proyecto final de carrera de Víctor Manuel Agüero Requena.

6.2. Editor

El editor es una aplicación que permitirá al usuario diseñar y construir los escenarios de juego, con un sencillo método, como puede ser arrastrando imágenes con el *mouse*. A continuación definiremos las herramientas que contiene este editor , pero antes observamos una panorámica de la aplicación en la figura 78.

Figura 78: Captura de la aplicación por defecto.

Esta es la primera escena que veremos al arrancar el editor, los diferentes módulos se encuentran apilados en la parte superior izquierda, pero podemos cambiarlos de posición arrastrándolos con el *mouse* y colocándolos a nuestro gusto.

Para simplificar la lógica de juego, esta aplicación estará dividida en tres capas, la capa de decoraciones donde las imágenes no interactuarán, la capa de elementos donde se colocarán todo tipo de elementos interactuables y la capa de fondos donde colocaremos la música y fondos de pantalla del juego.

La mayoría de la información que necesita el editor para trabajar se encuentra en la nube, esto quiere decir que necesitamos estar conectados a internet para poder trabajar con él. Cada vez que hagamos una modificación a algún fichero, el editor se comunicará con el servidor para informarle de los nuevos cambios.No se ve una comunicación síncrona a la vista del usuario, porque el editor utiliza una comunicación asíncrona para no interrumpir el trabajo, cuando por ejemplo modificamos la estructura de archivos como crear una carpeta nueva, el editor se comunica con el servidor y actualiza el fichero que controla esta estructura. Por lo tanto podemos trabajar diseñando escenarios desde cualquier parte del mundo, desde cualquier dispositivo que soporte un navegador, a cualquier hora y dejar los archivos en el editor que aunque apaguemos la aplicación se conservarán.En resumen el editor cubre los siguientes requisitos: **Rendimiento, disponibilidad, accesibilidad, usabilidad, estabilidad, portabilidad, operatividad, interoperabilidad, escalabilidad, concurrencia, mantenibilidad, interfaz.**

6.2.1. Herramientas

La barra de herramientas, es la barra que nos encontramos situada a la izquierda de la aplicación en posición vertical, manteniendo *click* izquierdo en la parte superior puede ser arrastrada. Esta barra es un contenedor de botones y cada uno de ellos tiene una función diferente. Podemos considerar para simplificar que cada herramienta ejecuta o activa una rutina diferente. A continuación se explica cada una de las herramientas.

6.2.1.1. Drag & Drop

El cursor con forma de mano, es una herramienta que se alterna con el lápiz o la goma, es decir que cuando activemos una de estas tres se desactivarán las dos herramientas restantes.

Su función cuando esté activada, será coger imágenes del explorador de archivos (véase 6.2.3. Explorador de archivos) y arrastrarlas hasta la posición deseada del escenario (véase 6.2.4. Escenario). También nos permite coger una imagen del mismo escenario y reubicarla en cualquier otra posición, siempre que intentemos coger una imagen de la capa que tenemos activada.

Cuando el usuario esté arrastrando una imagen al escenario y la suelte en la posición final, la propia aplicación calculará la posición más cercana al centro de la *tile* y centrará dicha imagen. Resumiendo el cálculo quedaría algo así:

```
fotoSeleccionada.posx = Math.floor(posicionx / 100);
fotoSeleccionada.posy = Math.floor(posiciony / 100);
```

Pasamos las coordenadas de ratón a coordenadas de escenario y cuando pintamos la imagen, la colocamos en las coordenadas almacenadas sumándole la mitad de la anchura de la imagen, para que quede centrada.

Figura 79. Cursor Mano

6.2.1.2. Lápiz

El cursor con forma de lápiz, es una herramienta que se alterna con la mano o la goma, es decir que cuando activemos una de estas tres se desactivarán las dos herramientas restantes.

Su función cuando esté activada, será seleccionar una imagen del explorador de archivos y clickar en la posición deseada del escenario, así aparecerá esta imagen en esa *tile*. Solo podremos clicar para pintar si tenemos la capa correcta seleccionada, es decir no podremos pintar una decoración en la capa de elementos.

Cuando el usuario clique con el lápiz en el escenario, la propia aplicación calculará la posición más cercana al centro de la *tile* y centrará la imagen que tenemos seleccionada(la imagen seleccionada queda reflejada en el cuadro de información véase 6.2.2. Cuadro de información). Para diferenciar entre las diferentes herramientas la aplicación guarda el estado en una variable de la siguiente forma:

```
variable cursor : String;
      si  boton.pulsado == “mano”
            entonces cursor = “mano”;
      sino si boton.pulsado == “lapiz”
            entonces cursor = “lapiz”;
      sino si boton.pulsado == “goma”
            entonces cursor = “goma”;
      fsi
```

Como en la herramienta anterior pasamos las coordenadas de ratón a coordenadas de escenario y cuando pintamos la imagen, la colocamos en las coordenadas almacenadas sumándole la mitad de la anchura de la imagen, para que quede centrada.

Figura 80. Cursor lápiz

6.2.1.3. Goma

El cursor con forma de goma, es una herramienta que se alterna con el lápiz o la mano, es decir que cuando activemos una de estas tres se desactivarán las dos herramientas restantes.

Figura 81. Cursor Goma

Su función cuando esté activada, será clicar en una *tile* del escenario y eliminarla, visualmente para el usuario, hay un cambio entre la decoración que había en la *tile* y una *tile* vacía.

Cuando el usuario clique una *tile* del escenario, la propia aplicación calculará la posición más cercana al centro de la *tile* y eliminará la imágen situada en esa posición de la siguiente forma:

```
escenario[Math.floor(posicionx / 100)][Math.floor(posiciony / 100)].imagen = tileVacia;
```

Pasamos las coordenadas de ratón a coordenadas de escenario, localizamos la imagen en el escenario y la cambiamos por una imagen vacía.

6.2.1.4. Capa Decoración

El escenario está dividido en tres capas, también conocidas como mundos, el mundo físico, mundo visual y *background*. Este es el mundo visual, la capa **decoración**:

Figura 82. Icono capa decoración

Cuando cliquemos en la capa de decoración, se desactivarán el resto de las capas y esto quiere decir que estamos trabajando con la capa de decoración, esto significa que todas las imágenes que coloquemos en el escenario, quedarán guardadas como decoración.

Cuando seleccionamos una capa el botón queda marcado oscurecido y el fondo del navegador cambia a ser de color verde, para que rápidamente nos situemos en que capa estamos trabajando.

Las *tiles* que pintemos con esta capa activa se guardarán en una matriz similar a la del escenario. Es una matriz de NxM casillas y en cada una de ellas introducimos una imagen sin ningún tipo de comportamiento. Para rellenar esta capa tenemos que hacer uso de las herramientas explicadas en los tres apartados anteriores.

Figura 83. Switch Capas I

Las imágenes que colocaremos en esta capa serán aquellas que no tienen ninguna función lógica, solo **decorar** o **colisionar**, por ejemplo las *tiles* que hacen la función de suelo o pared, también algún arbusto como decoración.

Figura 84. Ejemplos de decoración

6.2.1.5. Capa Elementos

Este es el mundo físico, la capa **elementos:**

El fondo del navegador se pintará de color rojo. Aquí trabajamos con los elementos lógicos y físicos animados.

La Animación física por computadora o Física de juego es un tipo de programación, donde supone la introducción de las leyes de Física en un simulador o motor de juego. El propósito es hacer que los efectos físicos de los objetos creados o modelados tengan las mismas características que en la vida real, teniendo en cuenta, por ejemplo, gravedad, masa, fricción, restitución, etc.

Figura 85. Switch Capas II

La simulación de la física en la programación es solo una aproximación cercana a la física real (si bien se acerca mucho a la física que tendría un objeto en realidad, no es igual de realista que en la realidad), y el cómputo es desarrollado usando valores discretos. Hay muchos elementos que forman los componentes de la simulación física.

La capa de elementos es la que contiene todos aquellos objetos con algún tipo de comportamiento, ya sea para que el jugador lo controle, una inteligencia artificial o un tipo de física como puede ser la gravedad.

En ella colocaremos aquellos elementos como pueden ser enemigos en el juego piezas que moveremos, sensores o personajes:

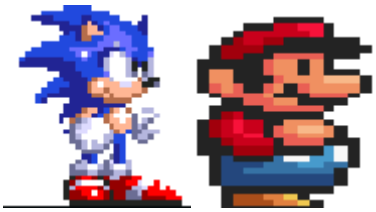


Figura 86. Ejemplo de elementos

Estos elementos, solo los podremos modificar, si tenemos la capa elementos activa y no otra.

Figura 87. Icono capa elementos

6.2.1.6. Capa Backgrounds

Este es el mundo *background*, la capa ***background*:**

El fondo del navegador se pintará de color azul. En esta capa es donde introducimos la música del juego, y las imágenes decorativas del horizonte de los juegos. Como extra en este punto se podría implementar el *paralax* un elemento muy importante en los juegos de plataformas.

Desplazamiento diferencial es una técnica (de uso frecuente en las consolas y ordenadores de 16 bits), que tiene como objetivo dar una impresión de profundidad a través de porciones del escenario desfilando a velocidades ligeramente diferentes. Se puede hablar de perspectiva de movimiento.

Figura 88. Switch Capas III

Esto es en realidad una adaptación de una técnica ampliamente utilizada por los dibujos animados, que consiste en hacer desfilar celuloides a velocidades diferentes.
Así, un escenario utilizando cinco desplazamientos diferenciales se verá dividido en cinco partes, la parte más baja (más lejos del horizonte, excepto en los casos de la capa de primer plano) se desplazará más rápido. La porción superior desplazará ligeramente más lenta, y la de encima aún más lenta, etc. (o viceversa).

En el caso de este editor el nivel de profundidad es de dos porque solo podemos mover la capa principal y el fondo de la pantalla.

Figura 89. Icono capa de fondo

6.2.1.7. Cargar

La herramienta cargar es lo que nos servirá para recuperar un escenario guardado anteriormente en un fichero.

Dado un fichero con extensión "*.json*" lo leeremos con *ajax* y actualizaremos las variables del entorno de la aplicación con las que nos encontramos en el fichero. El contenido del fichero tiene que tener el formato que utiliza la aplicación aunque este fichero no haya sido escrito por la herramienta de guardado. Los ficheros también pueden ser escritos a mano ya que no tienen ningún tipo de cifrado, pero sería mucho más incómodo.

A continuación un pequeño *snippet* de como funciona ajax para leer los ficheros y actualizar el estado del editor:

```
$.ajax({
        type: "GET",
        url: "json/escenario.json",
        dataType: "json",
        success: function(json) {
            escenario = new Escenario();
            document.getElementById('title').innerHTML = json['titulo'];
            ...
```

Figura 90. Botón de cargar

6.2.1.8. Guardar

Esta es la herramienta que se encarga de almacenar el estado actual de nuestro escenario en un fichero donde el usuario quiera guardarlo.

Para esta herramienta he tenido que utilizar una librería llamada **blob,** se encarga de gestionar la descarga de ficheros, así el navegador permitirá al usuario descargar ficheros utilizando la configuración de descarga de su navegador.

Antes de almacenar en un fichero con extensión *".json"* hay que adaptar la información que tenemos en el escenario al fichero, simplemente es poner etiquetas para que al leer el *json* esta información se identifique.

A continuación un pequeño *snippet:*

```
function guardaJSON() {
	var matrizJSON = new Array();
	var elementosJSON = new Array();

	transformarMatrizAJSON(matrizJSON);
	transformarElementosAJSON(elementosJSON);

	salida = {};
	salida.titulo = document.getElementById("tituloJuego").value;
	salida.canvasAncho = canvas.width;
	salida.canvasAlto = canvas.height;
	salida.matriz = matrizJSON;
	salida.listaElementos = elementosJSON;

	var blob = new Blob([JSON.stringify(salida)], {type: "text/plain;charset=utf-8"});
	saveAs(blob, "mapa.json");
}
```

Guardo en la variable "salida" todo lo que quedará almacenado en ese fichero, que por defecto se llama "mapa.json".

Figura 91. Botón de guardar

6.2.1.9. Play

Por último el botón *play,* es la herramienta que nos permite probar nuestro escenario. Cuando accionemos este botón, la aplicación automáticamente cambiará de editor a juego en ejecución.

Para ello lo que hace el editor, es enviar un archivo temporal al servidor, para que cuando se esté ejecutando el juego sepa de donde cargar el escenario. Por lo tanto cuando se ejecute el juego en modo *"Play"* siempre pedirá al servidor este archivo temporal que previamente ha guardado el editor.

Cuando el editor envía este archivo temporal al servidor la función es similar a la función de guardar y cargar, primero guardamos el mapa y después tenemos que utilizar *AJAX* para comunicarnos con el servidor, decirle donde está y desde donde se ejecuta. Para comunicarnos con el servidor podemos observarlo en el siguiente *snippet:*

```
$.ajax({
	type: "POST",
	url: RELATIVE_URL + "/guardaMapa",
	data: json,
	contentType: 'application/json; charset=utf-8',
	success: function (datos) {
		if (datos) {
		//aqui ha guardado el mapa que vamos a lanzar
			window.location.href = "motor/public_html/index.html" + '?v='+new
			Date().getTime();
		}
		else {
			alert("No se encuentra el servidor");
		}
	}
});
```

Figura 92. Botón para probar el escenario

6.2.2. Cuadro de Información

El cuadro de información es el cuadro que encontraremos en la parte superior izquierda junto al cuadro de herramientas. Este cuadro es el que utilizaremos para obtener información como la imagen que estamos tratando en el cuadro de imagen situado en su interior a la izquierda, el tipo de elemento en el desplegable, las coordenadas en las que se encuentra el elemento que estamos tratando y todo tipo de información que necesite cada elemento.

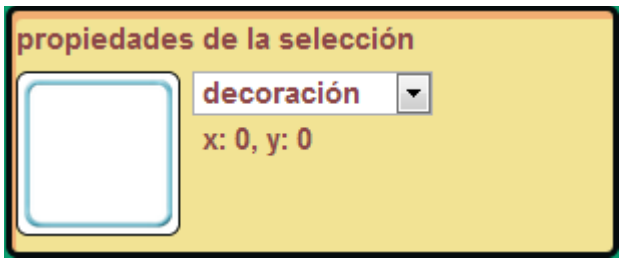


Figura 93. Bloque de propiedades de selección

6.2.2.1. Tipo

El tipo de un elemento sirve para determinar su comportamiento en el escenario, también su posición en las capas en las que está dividido el editor.

En el editor encontraremos un desplegable con una opción seleccionada por defecto que es decoración, esto significa que el elemento que tengamos seleccionado(por defecto *tile* vacía) será de tipo decoración y solo será una imagen. Para asignar un comportamiento a un elemento seleccionado, debemos seleccionar el elemento y después en el desplegable seleccionar el comportamiento que le queramos asignar. Si el comportamiento que le asignamos es un comportamiento que pertenece a otra capa del editor en la que no estamos trabajando, automáticamente el elemento pasará a estar en la nueva capa. Por ejemplo, si cogemos un elemento que inicialmente es del tipo "**decoración**" y le asignamos que ahora es del tipo "**mario**", como "mario" pertenece a la capa elementos y no a la capa decoración, el elemento se cambiará de capa y si no cambiamos de capa no podremos modificarlo.

Para añadir una nueva propiedad al desplegable del cuadro de información, tendremos que añadir una nueva entrada en el HTML, como extra externo al proyecto se podría incluir esta funcionalidad dentro de un botón integrado en el cuadro. El comportamiento que se ejecutará en el juego, está programado en el motor de juego, cada tipo de propiedad pertenecerá a una clase programada dentro del motor de juego.

Inicialmente en el editor hemos programado las clases que consideramos indispensables, no obstante siempre queda la posibilidad de expandirlo con nuevas clases y nuevos comportamientos, a continuación una breve descripción de las clases de elementos que encontraremos en el editor por defecto:

- **decoración:** Cuando asignamos la propiedad decoración a una imagen, esta imagen no tiene ningún tipo de interacción ni física, solo aparecerá la imagen en el juego y no se moverá.

- **mario:** Esta clase tiene el nombre en honor a Mario Bross, porque la imagen que adquiera esta propiedad en el juego se comportará como un juego de plataformas al estilo Super Mario Bross (véase 3.4.1. Plataformas). Si generamos un escenario, con los suficientes suelos con colisiones y le añadimos un elemento con la propiedad "mario" y le damos a "*Play"* veremos que podemos controlar este elemento, con las flechas del teclado.

- **coche:** Es la clase que mueve un elemento como si fuera con coche teledirigido visto desde arriba, igual que el elemento anterior lo podremos mover con las flechas del teclado, y se moverá hacia donde el elemento esté mirando inicialmente, calculará colisiones totales y colisiones manuales.

- **clicker:** Esta clase funciona clicando con el botón izquierdo del ratón, de ahí viene su nombre. Básicamente si a una imagen le asignamos la propiedad "clicker" dentro del juego utilizará un "pathfinding" para moverse, calculará el camino más corto evitando pasar por casillas con colisión y se desplazará al lugar donde el jugador haya clicado.

- **enemigo:** Es un enemigo patrulla que se desplazará de izquierda a derecha hasta que encuentre una colisión para volver a caminar en dirección contraria.

- **colisiónTotal:** Cuando una casilla tiene colisión total significa que todos los elementos anteriores colisionarán con esta y no podrán atravesarla.

- **colisiónManual:** Una casilla con colisión manual es aquella que no colisiona por toda la *tile* sino solo por la parte que seleccionemos con los sliders que explicaremos a continuación en el siguiente apartado.

Todas estas clases están más detalladas en la documentación de Victor Manuel Agüero entregada junto a esta.

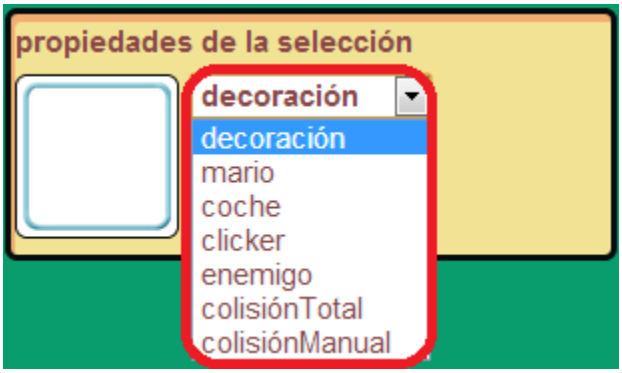


Figura 94. Desplegable propiedades de selección

6.2.2.2. Sliders

Los *Sliders* son las herramientas que utilizaremos para decidir como colisiona una colisión manual, o dicho de otra forma una colisión parcial que no ocupa toda la *tile* en su totalidad.

Si seleccionamos la propiedad colisión total en un elemento, significa que el personaje que llegue a esta casilla colisionará con la *tile* entera, pero las imágenes no siempre encajarán perfectamente con la forma cuadrada de una *tile*, pero para eso tenemos la colisión manual. En la colisión manual, tendremos que especificarle al editor con unos *sliders* de donde a donde colisionará.

Figura 95. Ejemplo de colisión con sliders

En la imagen vemos los dos *sliders* uno magenta y otro cian, que podremos arrastrar con el cursor para indicarle al editor la inclinación con la que queremos que colisionen nuestros personajes.

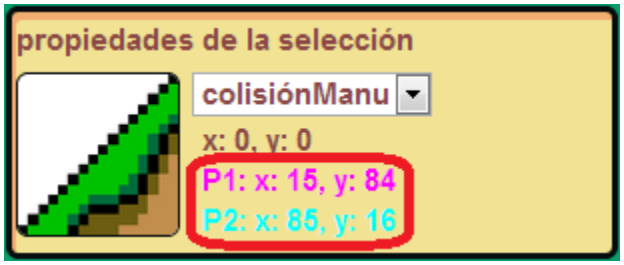


Figura 96. Posición de los sliders

Para más precisión, en el cuadro de información nos muestra dónde están las coordenadas relativas a la casilla de cada *slider*.

6.2.2.3. Coordenadas

Las coordenadas de cada elemento podremos verlas junto a la foto, para ahorrarnos tener que contarlas para saber donde están. Como extra externo al proyecto podría incluirse que esa información también sea modificable para trasladar los elementos a una nueva ubicación.

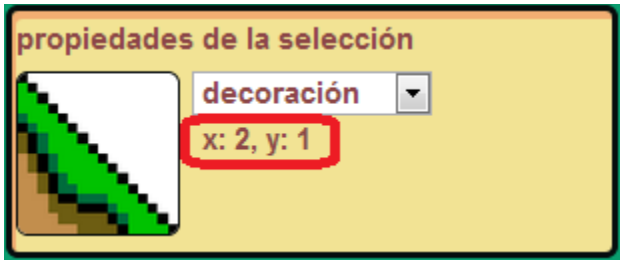


Figura 97. Coordenadas de una casilla

6.2.3. Explorador de archivos

El explorador de archivos incluye las herramientas necesarias para gestionar los archivos que queramos incluir en el editor, los formatos soportados de estos archivos son: **JPG, PNG, GIF**.

Figura 98. Explorador de archivos

Inicialmente en el explorador de archivos encontramos tres botones y un marco en el que podremos arrastrar imágenes desde nuestro sistema externo al navegador. Este sistema es muy

importante porque es la única forma que proporciona la aplicación de incluir nuevas imágenes en el editor.

Cada vez que queramos introducir un nuevo elemento, necesitaremos una imagen representativa, esta la introduciremos en el editor con el botón de nueva imagen.

6.2.3.1. Crear carpeta

Si queremos ordenar las imágenes de la barra del explorador de archivos, tenemos la opción de crear carpetas. La aplicación internamente almacena esta organización en un XML (véase 6.2.5.2. almacenamiento del explorador de archivos), por lo tanto al crear una carpeta nueva, este XML es modificado incluyendo una nueva línea en el nivel actual de la jerarquía de este XML, colocando una nueva carpeta identificada por orden, esto quiere decir que las carpetas no tienen un nombre sino una posición en la jerarquía de archivos.

Figura 99. Botón para crear carpeta

Para modificar este XML hay que acceder al servidor, ya que es el lugar donde está almacenado este fichero. Para comunicarse con el servidor hay que utilizar *ajax* y pedirle que introduzca una nueva línea en el fichero XML.

```
$.ajax({
        type:"POST",
        url: RELATIVE_URL + "/set/xml",
        dataType: "xml",
        success: function(xml) {
            $(xml).find("directorios > carpeta").each(function(i) {
                var img = new Image();
                img.src = "./js/images/carpeta.png";
                fotosMenu[i] = img;
                fotosMenu[i].id = $(this).attr("id");
                fotosMenu[i].tipo = "carpeta";
                fotosMenu[i].marcada = false;
                carpetas = i+1;
            });
            $(xml).find("directorios > foto").each(function(i) {
                var img = new Image();
                img.src = $(this).text();
                fotosMenu[i+carpetas] = img;
                fotosMenu[i+carpetas].tipo = "decoraciÃ³n";
                fotosMenu[i+carpetas].marcada = false;
            });
        },
        error: function(a) {
            alert(a.statusText);
        }
    });
```

En Resumen este *snippet* lo que hace es decirle al servidor que tiene que introducir una carpeta nueva en el XML y también actualiza la barra de imágenes del explorador de archivos que estamos viendo con la nueva barra actualizada que nos enviará el servidor cuando acabe de modificar el XML, si algo falla se informará al usuario con el error.

6.2.3.2. Volver a la carpeta principal

Para navegar entre carpetas y buscar las imágenes que hay dentro de ellas, solo hay que hacer dos clics sobre una carpeta. Al hacer el primer clic veremos que la carpeta se ilumina de amarillo, al segundo entraremos en ella y nos mostrará las imágenes y carpetas de su interior.

Cuando hayamos entrado en una o varias carpetas la única forma que hay de volver al principio de los ficheros es utilizar el botón de la figura 100. Cuando pulsemos el botón de volver a la carpeta principal, el sistema volverá a leer el XML que contiene la distribución de los archivos, pero solo leerá el primer nivel de la jerarquía y los pintará.

Figura 100. Botón de navegar a la carpeta principal

6.2.3.3. Adjuntar Nuevo Archivo

Para enviar archivos al servidor y así poderlos incluir en los escenarios, hay que subirlos al servidor. Cuando un archivo queda en el servidor, cualquier usuario puede verlo desde cualquier parte del mundo y así utilizarlo.

¿Cómo se sube una imagen por parte del usuario?. Para subir una nueva imagen, por parte de usuario, se puede pulsar el botón con la imagen de una cámara fotográfica, como el de la figura, y nos creará un diálogo para buscar nuestro archivo. Hay otra opción, arrastrar el archivo en el recuadro de puntos de alrededor del botón o también sobre el botón.

Figura 101. Botón de subir una nueva imagen

El tipo de envío que utilizaremos para enviar este archivo al servidor es un "*POST*", explicado en los protocolos de html, y para gestionar esta transacción, es necesario descargar la librería pública "***fileuploader***" que sirve para gestionar este envío con el servidor.

A continuación en un pequeño *snippet* observamos lo sencillo que es el uso de la librería:

```
$(document).ready(function()
{
	$("#fileuploader").uploadFile({
		type:"POST",
		url:"/upload",
		allowedTypes:"png,gif,jpg,jpeg",
	});
});
```

En este pequeño fragmento de código vemos que para asignar esta función a un objeto HTML solo hay que asignarle el *tag "fileuploader"* y la librería ya se encarga de gestionar el envío del archivo.

Una vez indiquemos la imagen que queremos subir al servidor el editor nos mostrará una barra de carga conforme está subiendo el archivo, una vez terminado podemos cerrarla y automáticamente aparecerá en la barra del explorador de archivos.

6.2.3.4. Extras

Como extra del proyecto sobre la barra de herramientas, se podría implementar la eliminación de carpetas y archivos por parte de usuario, ya que ahora es necesario el administrador para eliminar los archivos.

Figura 102. Imágenes de botones extra

También sería posible mejorar el uso de las carpetas con un identificador de nombre y que las imágenes puedan arrastrarse por el navegador de archivos, así pudiendo reordenarlas o introducirlas en carpetas una vez creadas en otro lugar.

6.2.4. Escenario

El escenario es una matriz en la que almacenaremos el estado de cada casilla. Se compone por una cantidad **N**x**M** casillas donde N es el número de casillas en horizontal y M el número de casillas en vertical. Para modificar este tamaño general del escenario, utilizaremos el botón :  que encontraremos en la parte inferior y en el lateral derecho del escenario, tanto como para extender el escenario en horizontal como para extenderlo en vertical. Pero si lo que queremos es reducirlo utilizaremos el botón:  que encontraremos en la parte superior y lateral izquierda del escenario.

Internamente el escenario, es una clase *array[array[]]* de *javascript*, en la que podemos introducir todo tipo de valores y tratarla como un diccionario. En ella introducimos un valor por casilla, en el que incluimos que imagen deberá pintar y que comportamiento utilizará, si este elemento necesita mas datos como en el caso de las colisiones manuales que necesitan las coordenadas de los sliders, el escenario también incluye esa información dentro de la casilla.

Por otro lado tenemos una *array[]* paralela al escenario que es la que incluye los elementos con interacción como puede ser un personaje jugable o un enemigo. Esto significa que en una misma casilla puede haber un elemento con interacción y una decoración, pero no dos decoraciones o dos elementos con decoración.

6.2.4.1. Drag & drop

Dentro del escenario podemos interactuar con las imágenes usando varias herramientas como está explicado anteriormente. Estas herramientas sirven para traer material del explorador de archivos, pero también nos sirven para mover imágenes del mismo escenario a una nueva ubicación del mismo.

En la figura podemos observar como la herramienta de la mano está arrastrando una *tile* con forma de suelo a una nueva ubicación.

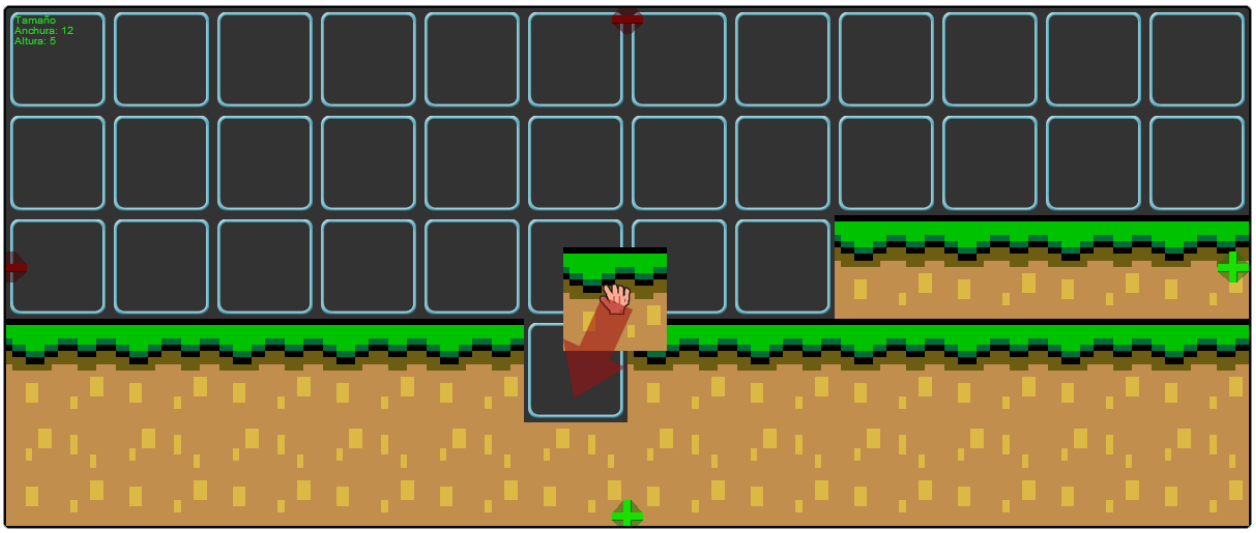


Figura 103. Captura arrastrando decoración

6.2.4.2. Sliders Drag & drop

Cuando dentro del escenario tenemos elementos con otro tipo de interacción fuera de la común aparecen los nuevos actuadores, en este caso cuando queremos hacer una casilla con colisión manual, el escenario nos pinta la imagen de dos círculos que tendremos que colocar en una posición de la casilla para determinar la inclinación de esta casilla. Este tipo de colisión se aleja levemente de lo que es un videojuego basado en *tiles* ya que un videojuego basado en *tiles* la

interacción es con la *tile* entera y no con una porción de esta, pero como la mayoría de los videojuegos basados en *tiles*, lo utilizan hemos decidido implementarlo.

En la siguiente figura podemos observar como el cursor está arrastrando uno de los dos actuadores para determinar la inclinación de la colisión de esta *tile*.

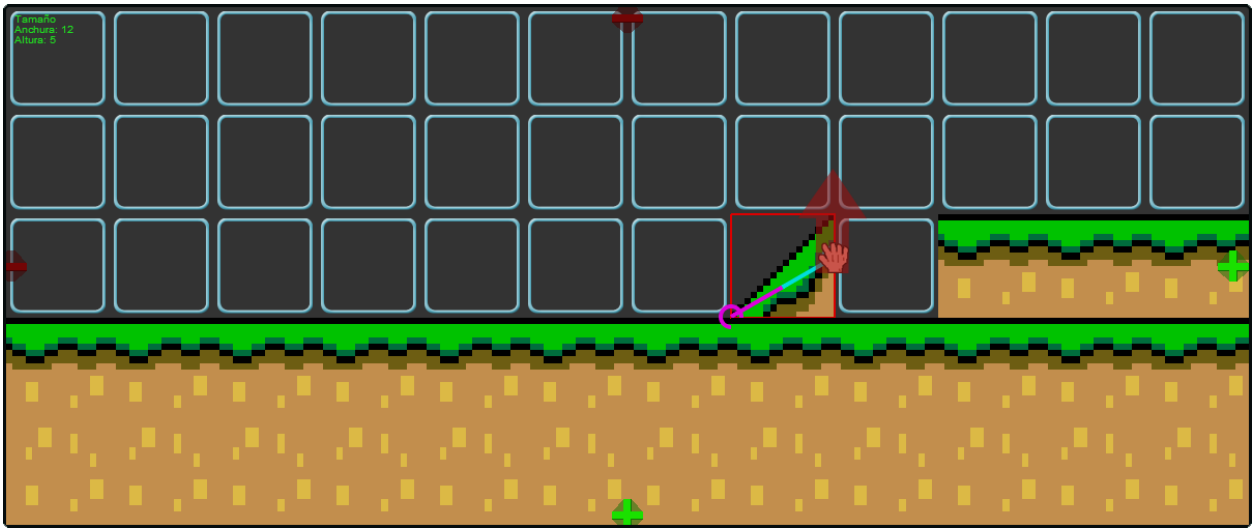


Figura 104. Captura modificando la posición de un slider

6.2.4.3. Crear

Para crear imágenes dentro de un escenario vacío, primero debemos determinar lo que medirá el escenario aunque este tamaño podrá ser modificado posteriormente. Una vez hemos establecido la altura y anchura del escenario podemos crear nuevos elementos utilizando las dos primeras herramientas (mano y lápiz). Con la mano podemos arrastrar una imágen directamente del explorador de archivos a la casilla deseada del escenario, o también del mismo escenario a una nueva posición del mismo. Con el lápiz, seleccionaremos una imagen en el explorador de archivos, y clicaremos en la casilla deseada (esta es la herramienta que deberemos usar si estamos trabajando desde un dispositivo **móvil** o **tablet**, ya que no nos permiten arrastrar, porque con el dedo se desplazará todo el contenido del navegador).

6.2.4.4. Eliminar

Para eliminar un elemento que ya hemos colocado en el escenario, deberemos seleccionar la capa en la que este elemento está colocado y después eliminarlo de dos formas, utilizando la herramienta goma y clicando sobre el elemento que deseamos eliminar, o arrastrando el elemento con la herramienta "mano" al exterior del escenario.

En la figura podemos observar como se elimina un elemento con la herramienta mano:

Figura 105. Eliminando una decoración arrastrando al exterior

6.2.5. Almacenamiento

El editor como es una aplicación web no puede tener datos persistentes, solo es capaz de mantener en el navegador datos temporales como la memoria caché o las *cookies*. Para solucionar esto hemos creado dos tipos de datos persistentes, los diferentes escenarios que almacenaré en ficheros con extensión *.json* y la información de las imágenes pintadas en el explorador de archivos que se guardan en otro archivo diferente con extensión *.xml,* ya que estas imágenes pertenecen al editor y no al escenario.

A continuación comentaré como almacenamos los dos tipos de datos.

6.2.5.1. Escenario

En el caso del escenario, hemos decidido almacenarlo en ficheros con el formato JSON, esto es porque como este escenario será leído por el mismo editor, o por el *game engine,* y ambos están programados en *javascript* JSON permite una lectura directa desde *javascript* como si este fichero ya fuera una variable. Por razones de comodidad hemos decidido que los escenarios se guarden en JSON, y en estos ficheros solo guardamos los datos necesarios, que serán, el título del juego,

el tamaño del escenario a pintar y el elemento que hay en cada coordenada de la matriz que compone el escenario.

Un fichero JSON generado por este editor contendría por ejemplo los siguientes datos:

{"titulo":"título del juego",
"canvasAncho":400,"canvasAlto":300,
"matriz":[[{"src":"http://caqui:5000/js/images/tile.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/tile.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sueloMario/tilesueloCentro.PNG","tipo":"decoración"}],
[{"src":"http://caqui:5000/js/images/tile.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/coches/marioANIMATION.PNG","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sueloMario/tilesueloCentro.PNG","tipo":"decoración"}],
[{"src":"http://caqui:5000/js/images/tile.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sonic.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sueloMario/tilesueloCentro.PNG","tipo":"decoración"}],
[{"src":"http://caqui:5000/js/images/tile.png","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sueloMario/suelom45Grados.PNG","tipo":"decoración"},
{"src":"http://caqui:5000/js/images/sueloMario/tilesueloCentro.PNG","tipo":"decoración"}]]
,"listaElementos":[]}

Si recargamos ese fichero con el editor o con el juego, pintará el siguiente escenario:

Figura 106. Resultado de generar el JSON anterior

6.2.5.2. Explorador de archivos

En el caso de las imágenes que almacenamos en el servidor, para pintar en el explorador de archivos y así poderlas introducir en los escenarios. He decidido que es mejor utilizar archivos con extensión XML, porque este archivo va a ser leído por el servidor y por el editor, como el servidor proporciona más facilidad de lectura y escritura sobre un XML este es el formato en el que generaré los archivos del explorador de archivos.

Estos XML se van generando y actualizando automáticamente cuando vamos subiendo nuevas imágenes al servidor, es en el servidor donde coge el actual XML y lo modifica con las nuevas direcciones de las nuevas imágenes.

Si existiera la opción de almacenar varios usuarios en este servidor con diferentes configuraciones para el explorador de archivos cada uno tendría que tener un XML asociado.

Un fichero XML generado por el servidor contendría por ejemplo los siguientes datos:

```
<Directorios>
	<carpeta><nombre>nombreCarpeta</nombre>
		<foto>./js/images/sueloMario/tilesueloCentro.PNG</foto>
		<foto>./js/images/sueloMario/tilesueloCentroBajo.PNG</foto>
	</carpeta>
	<carpeta><nombre>nombreCarpeta</nombre>
		<foto>./js/images/sueloMario/tilesueloCentro.PNG</foto>
		<foto>./js/images/sueloMario/tilesueloCentroBajo.PNG</foto>
	</carpeta>
	<foto>./js/images/sueloMario/tilesueloCentro.PNG</foto>
	<foto>./js/images/sueloMario/tilesueloCentroBajo.PNG</foto>
	<foto>./js/images/sueloMario/tilesueloCentro.PNG</foto>
	<foto>./js/images/sueloMario/suelo45Grados.PNG</foto>
	<foto>./js/images/sueloMario/suelom45Grados.PNG</foto>
	<foto>./js/images/sueloMario/suelo45GradosInv.PNG</foto>
	<foto>./js/images/sueloMario/suelom45GradosInv.PNG</foto>
	<foto>./js/images/coches/car.PNG</foto>
	<foto>./js/images/coches/car.PNG</foto>
</Directorios>
```

Cada vez que iniciamos la aplicación de nuevo el editor pide al servidor el XML que contiene el estado actual del explorador de archivos y el servidor le devuelve el XML al editor. Si el editor recibe el XML del ejemplo anterior nos pintaría el siguiente explorador de archivos:

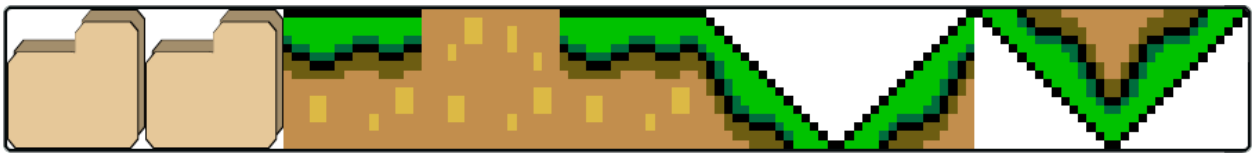


Figura 107. Resultado del explorador de archivos al leer el anterior XML

6.2.6. Comunicación con motor

Para comunicar el servidor con el motor tenemos dos opciones, la opción **manual** y la opción **automática**.

- **Manual:** Si lo que queremos es diseñar uno o varios escenarios con el editor y que el *game engine* los procese y ejecute, podemos crear el nuevo escenario y grabarlo en un fichero, el *game engine* por defecto buscará un fichero, también se puede crear un pequeño menú para que busque varios dependiendo de alguna acción. En resumen, tenemos que descargar tantos archivos como mapas queramos que contenga nuestro juego.

- **Automática:** La opción automática es utilizar el botón play, el *game engine* que tenemos alojado en el servidor, por defecto buscará un fichero, que el editor se encargará de enviárselo al servidor y de llamar al motor para que lo busque.

6.2.7. Servidor

Para este proyecto aparte de implementar un editor de videojuegos, también ha sido necesaria la implementación de un servidor que pueda gestionar todos los datos que maneje el editor. Para el servidor he decidido usar un *framework* llamado *flask* explicado en uno de los puntos anteriores. Es un *framework* muy sencillo, que se programa en *python*, de utilizar y por eso en este punto explicaré su uso como si fuera un tutorial, porque lo implementado en este proyecto sirve para gestionar todo tipo de aplicaciones.

Primero empiezo explicando los requisitos que debemos cumplir para hacer que nuestro pc sea capaz de ejecutar este servidor. Necesitamos tener instalado en nuestro sistema, python y flask, se pueden encontrar en las dos páginas oficiales.

Una vez instalados python y flask, solo tenemos que ejecutar el archivo en el que tengamos programado el servidor de la siguiente forma:

Figura 108. Captura de cmd ejecutando el servidor

A continuación explicaré las funciones que he tenido que implementar para mi proyecto y como las he implementado. Introduciré todo el código necesario para hacer que este servidor funcione correctamente.

6.2.7.1. Flask

Para que el servidor arranque satisfactoriamente deberemos tener una función "main", que puede estar vacía porque es lo que se ejecutará continuamente.

Primero importamos las librerías necesarias para poder recibir archivos y peticiones.

```
from flask import Flask
from flask import json
from flask import jsonify
import os
import xml
from xml.dom.minidom import Document
from xml.dom.minidom import parse, parseString
from flask import request, redirect, url_for
from werkzeug import secure_filename
```

Después declaramos nuestras variables de entorno que necesitemos.

```
path = os.path.dirname(__file__) + "\..\pfcHTML5\public_html"
UPLOAD_FOLDER = path+'\js\images'
```

```
UPLOAD_MAP = path+'\js\maps'
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif','json'])
```

A continuación configuramos el servidor con las rutas que necesitemos acceder.

```
app = Flask(__name__, static_folder=path, static_url_path="")
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['UPLOAD_MAP'] = UPLOAD_MAP
```

Definimos a qué directorio nos enviará el servidor cuando escribamos una dirección raíz como por ejemplo: http://www.ejemplo.com, en este caso he decidido que el servidor redireccione al usuario a index de mi aplicación que es la pantalla principal del editor.

```
@app.route("/", methods=['PUT', 'GET'])
def hello():
  if request.method == 'GET':
    return redirect('index.html')
  elif request.method == 'PUT':
    return "Error, no es posible enviar una petición PUT a esta dirección"

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
```

Creamos la función main que se ejecutará cuando el servidor se inicie. En este caso tengo el modo debug activado para consultar los mensajes.

```
if __name__ == "__main__":
    app.run(debug = True, host = "0.0.0.0")
```

La siguiente función sirve para actualizar todo tipo de xml, en este caso busca en el servidor un xml con el nombre "imagenes.xml", lo lee y le introduce un nuevo campo, en este caso al campo le llamo foto porque mi editor trabaja con imágenes pero puede utilizarse cualquier nomenclatura.

```
def actuXML(nombreFoto):
    fname = path + "\js\images\imagenes.xml"
    file = open(fname, 'r')
    data = file.read()
```

```
    file.close()
    datasource = open(fname)
    doc = parse(datasource)   # parse an open file
    top_element = doc.documentElement
    text = doc.createElement("foto")
    a = doc.createTextNode(nombreFoto)
    text.appendChild(a)
    #text.setAttribute("id", "miid")
    #foto = doc.createElement("foto")
    #foto.appendChild(a)
    #text.appendChild(foto)
    top_element.appendChild(text)
    file = open(fname, 'w')
    #file.write(doc.toprettyxml())
    file.write(doc.toxml())
    #import pdb; pdb.set_trace()
    return doc.toxml()
```

A partir de aquí explicaré las funciones más concretas de mi proyecto pero no obstante son reusables para cualquier tipo de programa que trabaje con comunicación continua con servidor.

6.2.7.2. Get XML

La función *get xml* la utiliza el editor cada vez que se ejecuta o actualiza el explorador de archivos para ver el estado actual del explorador de archivos. Primero le indicamos la dirección en la que el editor tendrá que llamar para obtener esta información, en este caso la dirección sería http://www.ejemplo.com/get/xml, y el servidor envía al cliente el xml que hay en su ubicación: path + "\js\images\imagenes.xml".

```
@app.route("/get/xml")
def aaa():
    fname = path + "\js\images\imagenes.xml"
    datasource = open(fname)
    doc = parse(datasource)
    return doc.toxml()
```

6.2.7.3. Set XML

La función *set xml* en el caso de mi servidor lo que hace es crear una nueva carpeta para el explorador de archivos del editor. Internamente lo que hace el servidor es leer el xml actual introducir una nueva entrada en la jerarquía de carpetas con una nueva carpeta y devolverle el xml al editor.

```
@app.route("/set/xml", methods=['POST'])
def bbb():
    fname = path + "\js\images\imagenes.xml"
    file = open(fname, 'r')
    data = file.read()
    file.close()
    datasource = open(fname)
    doc = parse(datasource)   # parse an open file
    top_element = doc.documentElement
    text = doc.createElement("carpeta")
    text.setAttribute("id", "miid")
    top_element.appendChild(text)
    file = open(fname, 'w')
    #file.write(doc.toprettyxml())
    file.write(doc.toxml())
    #import pdb; pdb.set_trace()
    return doc.toxml()
```

6.2.7.4. Actualizar mapa

Como he comentado anteriormente el editor tiene una manera de comunicarse automáticamente con el motor. Para comunicarse con el *game engine*, editor y motor deben acceder al mismo escenario. Lo que hacen internamente es enviar el nuevo escenario por parte del editor, eliminar el mapa anterior por parte del servidor y guardar el nuevo mapa por parte del servidor también, finalmente el servidor guarda la actualización y contesta al editor, el editor llamará al *game engine* pidiéndole que ejecute el nuevo mapa.

```
@app.route('/guardaMapa', methods=['POST'])
def upload_map():
  if request.method == 'POST':
          fname = UPLOAD_MAP + "\mapaTEST.json"
          file = open(fname, 'w')
          file.write(request.data)
          return "OK"
```

6.2.7.5. Upload

El servidor tiene la opción de recibir imágenes desde cualquier parte del mundo y almacenarlas, para después poder proporcionarselas a todos los usuarios. Para eso el servidor espera que la función de llamada utilice el método POST, en caso de no ser así no se ejecutará la subida del archivo.

Una vez recibido el archivo que el editor nos ha enviado, llamaremos la función "actuXML" que actualizará el XML que utiliza el servidor con la nueva información que es el nombre de la foto recibida y su dirección en el servidor, finalmente retornará este nuevo XML para que el editor pueda actualizar su explorador de archivos con la nueva imagen.

```
@app.route('/upload', methods=['POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files['file']
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            return actuXML('./js/images/' + filename)
```

6.2.7.6. CrossDomain

Cross Domain es un seguro de dominios cruzados que necesita el navegador para permitir el envío de información de un servidor a un cliente y viceversa. Por eso necesita saber que el servidor dispone de un cross domain.

Una solución de dominios cruzados es un medio de aseguramiento de la información que proporciona la capacidad de acceder de forma manual, de forma automática o mediante una transferencia entre dos o más dominios diferentes. Se trata de sistemas de hardware y software que permiten la transferencia de información entre integrados dominios incompatibles en este caso entre **python** y **javascript** del navegador. Cross domain es distinto de los enfoques más rigurosos, porque es compatible con la transferencia que de otro modo sería excluida por los modelos establecidos de los ordenadores, la red y la seguridad de datos, por ejemplo, el modelo Bell-LaPadula y modelo Clark-Wilson.

```
@app.route("/crossdomain.xml")
def cors():
  xml = """<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-
policy.dtd">
<cross-domain-policy>
    <!-- Read this: www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html -->
    <!-- Most restrictive policy: -->
    <!-- <site-control permitted-cross-domain-policies="none"/>-->
    <!-- Least restrictive policy: -->
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="*" to-ports="*" secure="false"/>
    <allow-http-request-headers-from domain="*" headers="*" secure="false"/>
</cross-domain-policy>
  """
return xml
```

7. Casos de Uso

A continuación muestro los casos de uso más comunes en el editor, son los casos de uso que son un poco más complejos y que el usuario utilizará repetidamente durante su trabajo.

Caso de Uso: "Subir archivo"

Actor Principal: Customer.
Precondición: nada
trigger: El usuario quiere subir un archivo al sistema

Escenario principal:
1. El actor proporciona el archivo requerido.
2. El sistema valida que los datos introducidos sean correctos.
3. El sistema da de alta un nuevo archivo con los datos recibidos.
4. El sistema actualiza la tabla de datos y responde al usuario.

Extensiones:
2a. **Validación fallida:**
    2a.1. El sistema muestra los errores de validación que se han producido.
    2a.2. vuelve al paso 1 del escenario principal.

Dependencias: ---
Materiales de soporte: Servidor
Autor: Editor, Rodolfo
Escenarios de validación:

Uso 1: El usuario clica el botón o arrastra un archivo en el lugar indicado para subir imágenes. El sistema le muestra el proceso de la subida del archivo. El usuario espera a que se complete la subida del archivo. El sistema guardará el nuevo archivo en el sistema.

Uso 2: El usuario clica el botón o arrastra un archivo en el lugar indicado para subir imágenes. El sistema le muestra el proceso de la subida del archivo. El usuario cancela el proceso o falla la conexión. El sistema le avisa de que se ha producido un error.
Historia: Creación: 28/09/2013

Caso de Uso: "Crear carpeta nueva"

Actor Principal: Customer.

Precondición: nada

trigger: El usuario quiere crear una carpeta nueva en el sistema.

Escenario principal:

1. El actor clica el botón crear carpeta.
2. El sistema recibe y valida la petición.
3. El sistema da de alta una nueva carpeta en el sistema.
4. El sistema actualiza la tabla de datos y responde al usuario.

Extensiones:

2a. **Validación fallida:**

2a.1. El sistema muestra los errores de validación que se han producido.

2a.2. vuelve al paso 1 del escenario principal.

Dependencias: ---

Materiales de soporte: Servidor

Autor: Editor, Rodolfo

Escenarios de validación:

Uso 1: El usuario clica el botón. El sistema actualiza el estado del explorador de archivos. El usuario espera a que se complete la acción. El sistema devuelve el nuevo fichero de estado del explorador de archivo. El usuario ve como aparece una nueva carpeta en el explorador de archivos.

Uso 2: El usuario clica el botón. El sistema actualiza el estado del explorador de archivos. El usuario espera a que se complete la acción. Falla la conexión. El sistema le avisa de que se ha producido un error.

Historia: Creación: 28/09/2013

Caso de Uso: "Guardar escenario"

Actor Principal: Customer.

Precondición: nada

trigger: El usuario quiere guardarse el escenario actual en su sistema propio.

Escenario principal:

1. El actor clica el botón guardar.
2. El sistema recibe y valida la petición.
3. El sistema genera un nuevo archivo.
4. El actor descarga el archivo que contiene su escenario.

Extensiones:

2a. **Validación fallida:**

2a.1. El sistema muestra los errores de validación que se han producido.

2a.2. vuelve al paso 1 del escenario principal.

Dependencias: ---

Materiales de soporte: Servidor

Autor: Editor, Rodolfo

Escenarios de validación:

Uso 1: El usuario clica el botón. El sistema genera el archivo. El usuario espera a que se complete la acción. El sistema devuelve el nuevo fichero. El usuario descarga el archivo.

Historia: Creación: 28/09/2013

Caso de Uso: "Crear una imagen nueva en el escenario"

Actor Principal: Customer.
Precondición: nada
trigger: El usuario quiere crear una imagen nueva en el escenario.

Escenario principal:
1. El actor arrastra una imagen al escenario.
2. El sistema recibe y valida la acción.
3. El sistema actualiza la matriz del escenario.

Extensiones:
2a. **Validación fallida:**
    2a.1. El sistema no actualiza la matriz de escenario.
    2a.2. El sistema no pinta la imagen en el escenario.

Dependencias: ---
Materiales de soporte: ---
Autor: Editor, Rodolfo

Escenarios de validación:

Uso 1: El usuario arrastra una imagen al escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. El usuario ve como aparece una nueva imagen en el escenario.

Uso 2: El usuario pinta una imagen en el escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. El usuario ve como aparece una nueva imagen en el escenario.

Uso 3: El usuario arrastra una imagen al escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. Falla la conexión. El sistema no actualiza la matriz del escenario. El usuario no ve como se pinta la imagen en el escenario.
Historia: Creación: 28/09/2013

Caso de Uso: "Eliminar una imagen del escenario"
Actor Principal: Customer. **Precondición:** nada **trigger:** El usuario quiere eliminar una imagen del escenario. **Escenario principal:** 1. El actor clica con la goma o arrastra una imagen fuera del escenario. 2. El sistema recibe y valida la acción. 3. El sistema actualiza la matriz del escenario. **Extensiones:** 2a. **Validación fallida:** 2a.1. El sistema no actualiza la matriz de escenario. 2a.2. El sistema no elimina la imagen en el escenario. **Dependencias:** --- **Materiales de soporte:** --- **Autor:** Editor, Rodolfo **Escenarios de validación:** Uso 1: El usuario arrastra una imagen fuera del escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. El usuario ve como desaparece la imagen del escenario. Uso 2: El usuario clica con la goma en el escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. El usuario ve como desaparece la imagen del escenario. Uso 3: El usuario elimina una imagen del escenario. El sistema actualiza el estado de la matriz del escenario. El usuario espera a que se complete la acción. Falla la conexión. El sistema no actualiza la matriz del escenario. El usuario no ve como desaparece la imagen del escenario. **Historia:** Creación: 28/09/2013

El conjunto de casos de uso asociados al motor de juego se encuentran en la documentación de Victor Manuel Agüero Requena.

8. Costes y planificación

Como hemos trabajado en pareja, se ha necesitado utilizar un método de gestión y sincronización del trabajo, para organizarnos hemos decidido utilizar la metodología SCRUM, que ambos habíamos utilizado anteriormente.

Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

Aunque Scrum estaba enfocado a la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas: Scrum de Scrums.

Características de Scrum:
SCRUM es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el ScrumMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (interesados externos o internos), y el Team que incluye a los desarrolladores.

Durante cada sprint, un periodo entre una y cuatro semanas (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint. Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.

Scrum permite la creación de equipos autoorganizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan (a menudo llamado requirements churn), y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

Existen varias implementaciones de sistemas para gestionar el proceso de Scrum, que van desde notas amarillas "post-it" y pizarras hasta paquetes de software. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

Jeff Sutherland aplicó el modelo Scrum al desarrollo de software en 1993 en Easel Corporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation). En 1995 lo presentó junto con Ken Schwaber como proceso formal, también para gestión del desarrollo de software en OOPSLA 95. Más tarde, en 2001 serían dos de los promulgadores del Manifiesto ágil. En el desarrollo de software scrum está considerado como modelo ágil por la Agile Alliance.

La ficha adjunta incluye una descripción sinóptica del proceso y sus elementos que son:

- Roles: Propietario del producto, Gestor o Manager del Scrum, Equipo e Interesados.
- Componentes del proceso: Pila del producto (Product Backlog), Pila del sprint (Sprint Backlog), Incremento.
- Reuniones: Planificación del sprint, Revisión diaria, Revisión del sprint.
- Sprint

Para ayudarnos en la organización hemos utilizado una aplicación web, que ayuda a la coordinación de proyectos que utilizan el método SCRUM.

Trello es una aplicación basada en tarjetas y listas. Cada tarjeta representa una funcionalidad y cada lista un estado del proyecto. El tablera que nos proporciona trello nos permite plasmar el product backlog de nuestro proyecto.

En la siguiente figura podemos ver una captura de la aplicación con nuestro proyecto a mitad de desarrollo.

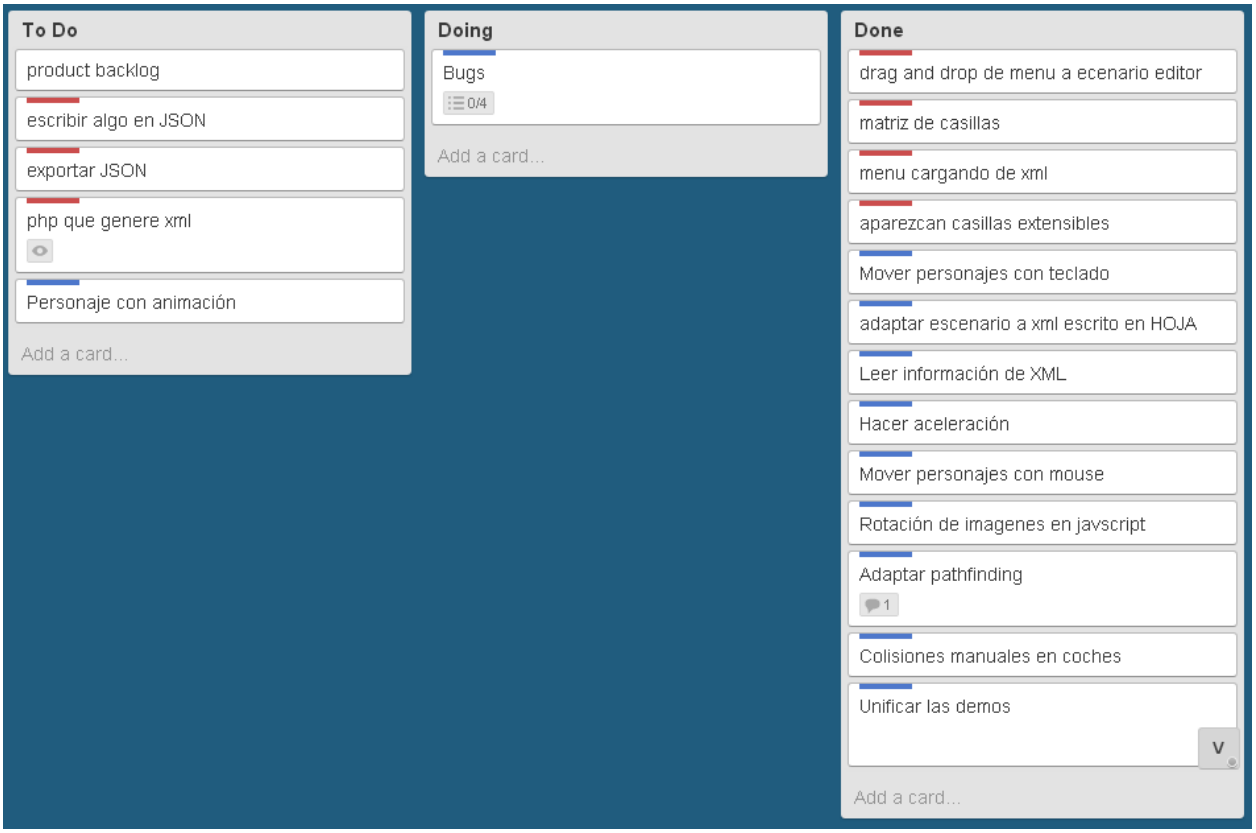


Figura 109. Captura de la aplicación Trello a mitad de proyecto

Aunque un Gantt no sea del todo compatible con la metodología SCRUM, hicimos uno al principio del proyecto, para hacernos una idea de la duración que necesitábamos para desarrollar el proyecto.

Game Engine	Descripción de la tarea	Fecha de inicio	Fecha de finalización	Progreso[1]	Trabajo	Recursos	Coste	Dias de trabajo[2]
1	**Dibujar escenario**	**25-Jun-2013**	**9-Jul-2013**	**100%**	**66**		**€ 1,320.00**	11
1.1	Leer JSON	25-Jun-2013	1-Jul-2013	100%	30		€ 600.00	5
1.2	Dibujar decoración	2-Jul-2013	6-Jul-2013	100%	24		€ 480.00	4
1.3	Dibujar personajes jugables	7-Jul-2013	9-Jul-2013	100%	12		€ 240.00	2
2	**Clicker**	**10-Jul-2013**	**15-Aug-2013**	**100%**	**162**		**€ 3,240.00**	27
2.1	Eventos de ratón	10-Jul-2013	15-Jul-2013	100%	24		€ 480.00	4
2.2	Movimiento	16-Jul-2013	20-Jul-2013	100%	24		€ 480.00	4
2.3	Pathfinding	21-Jul-2013	14-Aug-2013	100%	108		€ 2,160.00	18
2.4	Retoques estéticos	15-Aug-2013	15-Aug-2013	100%	6		€ 120.00	1
3	**Car**	**16-Aug-2013**	**7-Oct-2013**	**100%**	**222**		**€ 4,440.00**	37
3.1	Eventos de teclado	16-Aug-2013	19-Aug-2013	100%	12		€ 240.00	2
3.2	Movimiento adelante/atrás	20-Aug-2013	21-Aug-2013	100%	12		€ 240.00	2
3.3	Rotación	22-Aug-2013	25-Aug-2013	100%	12		€ 240.00	2
3.4	Retoques estéticos	26-Aug-2013	28-Aug-2013	100%	18		€ 360.00	3
3.5	Aceleración/Frenado	29-Aug-2013	30-Aug-2013	100%	12		€ 240.00	2
3.6	Colisiones	31-Aug-2013	9-Sep-2013	100%	36		€ 720.00	6
3.7	Colisiones Manuales	10-Sep-2013	7-Oct-2013	100%	120		€ 2,400.00	20
4	**Platform**	**8-Oct-2013**	**29-Nov-2013**	**100%**	**234**		**€ 4,680.00**	39
4.1	Movimiento	8-Oct-2013	9-Oct-2013	100%	12		€ 240.00	2
4.2	Colisiones	10-Oct-2013	18-Oct-2013	100%	42		€ 840.00	7
4.3	Salto	19-Oct-2013	22-Oct-2013	100%	12		€ 240.00	2
4.4	Gravedad	23-Oct-2013	28-Oct-2013	100%	24		€ 480.00	4
4.5	Colisiones manuales	29-Oct-2013	20-Nov-2013	100%	102		€ 2,040.00	17
4.6	Animaciones	21-Nov-2013	27-Nov-2013	100%	30		€ 600.00	5
4.7	Enemy	28-Nov-2013	29-Nov-2013	100%	12		€ 240.00	2
5	**Control táctil**	**30-Nov-2013**	**16-Dec-2013**	**100%**	**66**		**€ 1,320.00**	11
5.1	Eventos táctiles	30-Nov-2013	6-Dec-2013	100%	30		€ 600.00	5
5.2	Adaptación clicker	7-Dec-2013	10-Dec-2013	100%	12		€ 240.00	2
5.3	Adaptación car/platform	11-Dec-2013	16-Dec-2013	100%	24		€ 480.00	4
6	**Dibujar escenario en editor**	**25-Jun-2013**	**9-Jul-2013**	**100%**	**80**		**€ 1,600.00**	11
6.1	Dibujar escenario en editor	25-Jun-2013	27-Jun-2013	100%	40		€ 800.00	3
6.2	Pintar elementos en escenario	27-Jun-2013	3-Jul-2013	100%	20		€ 400.00	5
6.3	Escribir JSON	3-Jul-2013	9-Jul-2013	100%	20		€ 400.00	5
7	**Barra de herramientas**	**9-Jul-2013**	**24-Dec-2013**	**100%**	**245**		**€ 4,700.00**	121
7.1	Mano, lápiz, goma	9-Jul-2013	15-Jul-2013	100%	30		€ 600.00	5
7.2	capas	15-Jul-2013	31-Jul-2013	100%	40		€ 800.00	13
7.3	guardar	1-Aug-2013	30-Aug-2013	100%	80		€ 1,500.00	22
7.4	cargar	1-Aug-2013	30-Aug-2013	100%	80		€ 1,500.00	22
7.5	Botón "PLAY"	23-Dec-2013	24-Dec-2013	100%	15		€ 300.00	2
8	**Servidor**	**2-Sep-2013**	**5-Nov-2013**	**100%**	**320**		**€ 8,000.00**	47
8.1	Instalar y Configurar	2-Sep-2013	13-Sep-2013	100%	80		€ 2,000.00	10
8.2	Recibir XML	16-Sep-2013	23-Sep-2013	100%	80		€ 2,000.00	6
8.3	Tratar XML	24-Sep-2013	14-Oct-2013	100%	80		€ 2,000.00	15
8.4	Guardar y devolver XML	14-Oct-2013	5-Nov-2013	100%	80		€ 2,000.00	17
9	**Explorador de Archivos**	**5-Nov-2013**	**18-Dec-2013**	**100%**	**90**		**€ 1,800.00**	32
9.1	Crear carpeta	5-Nov-2013	11-Nov-2013	100%	30		€ 600.00	5
9.2	Subir archivo	11-Nov-2013	29-Nov-2013	100%	30		€ 600.00	15
9.3	Arrastrar Archivo y subir	29-Nov-2013	6-Dec-2013	100%	20		€ 400.00	6
9.4	Navegar por el explorador de archivos	6-Dec-2013	18-Dec-2013	100%	10		€ 200.00	9
10	**Cuadro información**	**23-Dec-2013**	**25-Dec-2013**	**100%**	**16**		**€ 320.00**	3
10.1	Pintar imagen	23-Dec-2013	24-Dec-2013	100%	8		€ 160.00	2
10.2	Pintar datos concretos	24-Dec-2013	25-Dec-2013	100%	8		€ 160.00	2
Total		**25-Jun-2013**	**16-Dec-2013**		**1501**		**€ 31,420.00**	**339**

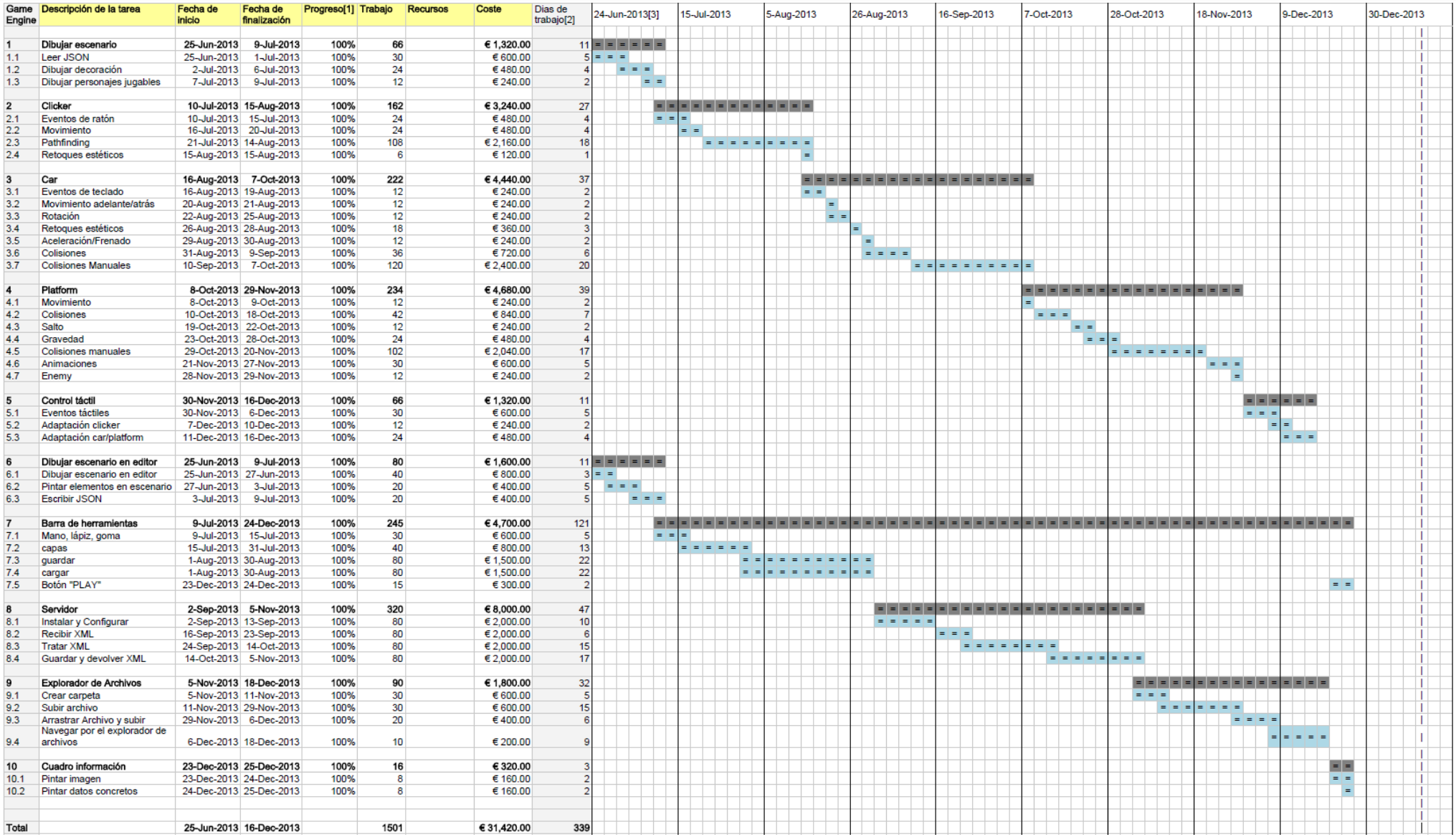


Figura 110. Diagrama de Gantt de los dos proyectos sincronizados

9. Conclusiones

- Nos ha servido para conocer las tecnologías web de última generación. Adentrarnos en el mundo del desarrollo de los videojuegos.
- Hemos notado, que para desarrollar un videojuego, no es necesario ningún don especial, sólo es necesario tener paciencia y dedicación.
- Los videojuegos para navegador tienen mucho potencial, pero HTML5 todavía está en desarrollo, los dispositivos que utilizan navegador todavía no soportan todas sus funcionalidades y las que sí utilizan, no las exprimen al 100%.
- Debido a que en el mundo de los videojuegos se puede imaginar casi cualquier cosa, se podrían ir añadiendo nuevas funcionalidades tanto al motor como al editor.
- Nuestro motor y editor son capaces de producir cualquier tipo de videojuego incluido en los géneros comentados, siempre y cuando se basen en *tiles*.

Actualmente el ordenador que estaba haciendo de servidor para permitir las transferencias entre las imágenes que aparecen en el editor y las que contiene el servidor es un pc de lsi, con la siguiente dirección (http://caqui.lsi.upc.edu:5000), si este servidor permanece encendido, se puede probar el proyecto con conexión a internet.

10. Bibliografía

Motor de videojuego (n.d.) En Wikipedia. Recuperado el 5 de diciembre de 2013, de http://es.wikipedia.org/wiki/Motor_de_videojuego

Historia de los videojuegos (n.d.) En Wikipedia. Recuperado entre el 4 y el 30 de diciembre de 2013, de http://es.wikipedia.org/wiki/Historia_de_los_videojuegos

History of videogames (n.d.) En Wikipedia. Recuperado entre el 4 y el 30 de diciembre de 2013, de http://en.wikipedia.org/wiki/History_of_video_games

Primer Videojuego. (n.d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, de http://es.wikipedia.org/wiki/Primer_videojuego

First Video Game. (n.d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, de http://en.wikipedia.org/wiki/First_video_game

Cathode Ray Tube Amusement Device. (n. d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://en.wikipedia.org/wiki/Cathode_ray_tube_amusement_device

OXO. (n.d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://en.wikipedia.org/wiki/OXO

Pilot Ace. (n. d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://es.wikipedia.org/wiki/Pilot_ACE

Spacewar! (n.d.) En Wikipedia. Recuperado el 4 de diciembre de 2013 en http://es.wikipedia.org/wiki/Spacewar!

Galaxy Game (n.d.) En Wikipedia. Recuperado el 5 de diciembre de 2013 en http://es.wikipedia.org/wiki/Galaxy_Game

Galaxy Game (n.d) En Computer History Museum. Recuperado el 5 de diciembre de 2013 en http://www.computerhistory.org/collections/catalog/102716148

Computer Space (n.d.) En Wikipedia. Recuperado el 5 de diciembre de 2013, en http://es.wikipedia.org/wiki/Computer_Space

Nolan Bushnell (n.d.) En Wikipedia. Recuperado el 5 al 10 de diciembre de 2013, en http://es.wikipedia.org/wiki/Nolan_Bushnell

Pong (n.d.) En Wikipedia. Recuperado el 5 de dicembre de 2013, en http://es.wikipedia.org/wiki/Pong

Gun Fight (n.d.) En Wikipedia. Recuperado el 9 de diciembre de 2013, en http://en.wikipedia.org/wiki/Gun_Fight

Death Race (n.d.) En Wikipedia. Recuperado el 9 de diciembre de 2013, en http://es.wikipedia.org/wiki/Death_Race

Breakout (n.d.) En Wikpedia. Recuperado el 9 de diciembre de 2013, en http://es.wikipedia.org/wiki/Breakout_(videojuego)

Adventure Game Interpreter (n.d.) En Wikipedia. Recuperado el 30 de diciembre de 2013, en http://es.wikipedia.org/wiki/AGI

SCUMM (n.d.) en Wikipedia. Recuperado el 30 de diciembre de 2013, en http://es.wikipedia.org/wiki/SCUMM

Unity3D Game Engine. (n. d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://en.wikipedia.org/wiki/Unity_(game_engine)

Unreal Engine. (n. d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://en.wikipedia.org/wiki/Unreal_Engine

RPG Maker. (n. d.) En Wikipedia. Recuperado el 4 de diciembre de 2013, en http://en.wikipedia.org/wiki/RPG_Maker

RPG Maker. (n. d.) En Phylomortis. Recuperado el 4 de diciembre de 2013, en http://web.archive.org/web/20040901070118/http://www.phylomortis.com/

Género de videojuegos. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://es.wikipedia.org/wiki/Género_de_videojuegos

Video games genres. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://en.wikipedia.org/wiki/Video_game_genres

Videojuego de aventura. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://es.wikipedia.org/wiki/Videojuego_de_aventura

Adventure game. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://en.wikipedia.org/wiki/Adventure_game

Colossal Cave Adventure. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://en.wikipedia.org/wiki/Colossal_Cave_Adventure

Mystery House. (n. d.) En Wikipedia. Recuperado el 1 de enero de 2014, en http://en.wikipedia.org/wiki/Mystery_House

Role-playing video game. (n. d.) En Wikipedia. Recuperado el 2 de enero de 2014, en http://en.wikipedia.org/wiki/Role-playing_video_game

Roguelike. (n. d.) En Wikipedia. Recuperado el 2 de enero de 2014, en http://en.wikipedia.org/wiki/Roguelike

Rogue (videogame). (n. d.) En Wikipedia. Recuperado el 2 de enero de 2014, en http://en.wikipedia.org/wiki/Rogue_(computer_game)

Massively Multiplayer online role-playing game. (n. d.) En Wikipedia. Recuperado el 2 de enero de 2014, en http://en.wikipedia.org/wiki/MMORPG

Strategy video game. (n. d.) En Wikipedia. Recuperado el 3 de enero de 2014, en http://en.wikipedia.org/wiki/Strategy_video_game

Advertising in video games (n. d.) En Wikipedia. Recuperado el 8 de enero de 2014, en http://en.wikipedia.org/wiki/Advergame

HUD (informática). (n. d.) En Wikipedia. Recuperado el 10 de enero de 2014, en http://es.wikipedia.org/wiki/HUD_(informática)

Cinemática (videojuegos). (n. d.) En Wikipedia. Recuperado el 11 de enero de 2014, en http://es.wikipedia.org/wiki/Cinemática_(videojuegos)

Jugabilidad. (n. d.) En Wikipedia. Recuperado el 11 de enero de 2014, en http://es.wikipedia.org/wiki/Jugabilidad

HTML5 (n. d.) En Wikipedia. Recuperado el 11 de enero de 2014, en http://en.wikipedia.org/wiki/HTML5

HTML Working Group (n. d.) En W3C. Recuperado el 11 de enero de 2014, en http://www.w3.org/html/wg/

Ajax (n. d.) En Wikipedia. Recuperado el 11 de enero de 2014, en http://es.wikipedia.org/wiki/AJAX

jQuery (n. d.) En Wikipedia. Recuperado el 11 de enero de 2014, en http://es.wikipedia.org/wiki/Jquery

jQuery, write less, do more. (n. d.) En jQuery. Recuperado el 11 de enero de 2014, en http://jquery.com/

Heurístics. From Amit's Thoughts on Pathfinding. Recuperado el 26 de diciembre de 2013, en http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Python. Recuperado el 8 de enero de 2014, en http://www.python.org/getit/

Flask. Recuperado el 8 de enero de 2014, en http://flask.pocoo.org/